

ENKBA-2.0

VAX 11/730 8085
TEST PART I

EP-ENKBA-DL-2.0
FICHE 1 OF 1

JAN 1983
COPYRIGHT © 82-83
MADE IN USA



IDENTIFICATION

Product code: ENKBA VERSION 2.0

Product name: 8085 BASED MICRO-DIAGNOSTIC FOR VAX-11/730 WCS
AND CPU MODULES

Product date: 11-OCT-82

Maintainer: BASE SYSTEMS DIAGNOSTIC ENGINEERING

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1982 by Digital Equipment Corporation. All Rights Reserved.

The following are trademarks of Digital Equipment Corporation.

DEC	DECsystem-10	DECSYSTEM-20
DECUS	MASSBUS	PDP
UNIBUS	VAX	VMS

digital

Table of contents

1.0	ABSTRACT	3
2.0	HARDWARE REQUIREMENTS	3
3.0	SOFTWARE REQUIREMENTS	3
4.0	PREREQUISITES	3
5.0	OPERATING INSTRUCTIONS	3
5.1	Options	3
5.2	Event Flags	4
5.3	APT Setup	4
6.0	PROGRAM FUNCTIONAL DESCRIPTION	4
6.1	Program Overview	4
6.2	Program Size	4
6.3	Program Run Times	4
6.4	Run-time Dynamics	4
6.5	Fault Detection	5
6.6	Performance During Hardware Failures	5
6.7	Program Applications	5
6.8	Test Descriptions	6
7.0	MAINTENANCE HISTORY	6

[illegible]

A more extensive WCS test is contained in the module ENKBF but because of its execution time (2 MIN 55 SEC) it was replaced with a less thorough test in this non-optional module. See the program listings for more details.

[illegible][illegible][illegible]

L

[illegible][illegible][illegible]

- [illegible]

[illegible][illegible][illegible][illegible][illegible]

22-
720
720
720
720

720
721
721
721

721
721
721
721
721

722
722
722

[illegible]

4

3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62

H 1

Fiche 1 Frame H1

Sequence 7

ENKBA 8085 based diagnostic for WCS/DAP module Rev 02.00

```

143 DATA AREA
308 PROGRAM ENTRY POINT
326 VARIABLES AREA
431 PROGRAM SPECIFIC SUBROUTINES
432 CON SET FOR_CO
451 CHECK RD
471 MAKE_NEXT_ADD
506 EXEC_INST
522 LD_CSR_INST
560 WCS_SIZER
636 WCS_SIZE_P
657 SHIFT_RT
683 LOAD_MOD_DAT
698 ERROR_4_TST
727 ERROR_4_2ND
751 TEST1_ERROR
769 TEST2_ERROR
835 TEST3_4_ERROR
865 CHECK_5_6_ERROR
883 TEST5_6_ERROR
915 TEST7_ERROR
941 TEST8_ERROR
973 TEST8_1_0
1002 TEST8_SHIFT
1035 TEST 1 - SERIAL SHIFT OF THE UPC
1238 TEST 2 - SERIAL SHIFT OF THE CSR
1573 TEST 3 - UPC NEXT ADDRESS TEST
1690 TEST 4 - UPC TEST WITH JUMP INSTRUCTION
1847 TEST 5 - WCS RAM DATA TEST
2176 TEST 6 - ABBREVIATED RAM MARCH PATTERN TEST
2351 TEST 7 - CONTROL STORE PARITY TEST
2637 TEST 8 - 2901 DATA PATH TEST (D TO Y)

```

725
725
725
725
725
725
725
725
725
725
725
725
725
725
725
726
726
726
726
726
726
726
727
727
727
727
727
727
727
728
728
728
728
728
728
728
729
729
729
729
729
72A
72A
72A
72A
72A
72A

7000	63
7000	64
7000	65
7000	66
7000	67
7000	68
7000	69
7000	70
7000	71
7000	72
7000	73
7000	74
7000	75
7000	76
7000	77
7000	78
7000	79
7000	80
7000	81
7000	82
7000	83
7000	84
7000	85
7000	86
7000	87
7000	88
7000	89
7000	90
7000	91
7000	92
7000	93
7000	94
7000	95
7000	96
7000	97
7000	98
7000	99
7000	100
7000	101
7000	102
7000	103
7000	104
7000	110
7000	111
7000	112
7000	113
7000	114
7000	115
7000	116
7000	117
7000	118
7000	119
7000	120
7000	121
7000	122
7000	123
7000	124
7000	125
7000	126
7000	127

.TITLE 'ENKBA 8085 based diagnostic for WCS/DAP module Rev 02.00'

COPYRIGHT (c) 1982 BY
DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS. ALL RIGHTS RESERVED.

```

: THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
: ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION
: OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER COPIES THEREOF
: MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON. NO
: TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.

```

```

; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE, AND
; SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

```

```

; DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
; SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

```

```

:++
: FACILITY:      VAX-11/730 MICRO-DIAGNOSTIC.

```

```
; ABSTRACT: This 8085 based diagnostic tests the hardcore logic of the
; WCS and DAP modules.
```

; ENVIRONMENT: ENKAA Micro-diagnostic Monitor

: AUTHOR: Jim Klumpp 7-JUN-81

```

: MODIFIED BY: David Mayo      8-MAR-82      VERSION 01.00
:
:               Ernie Preisig  19-AUG-82      VERSION 02.00 (RD CHANGES)
:

```

Mike Ricchetti 6-OCT-82
Add second error to Test 1. Shortened code (via subroutines)
to make room for modifications.

[illegible]

4

111

Fiche 1 Frame 11

Sequence 11

:ADDITIONAL DATA

ZZ-
73A
73A
73A
73A
73A
73A
73B
73B
73B
73E
73B
73E
73E
73B
73E
73B
73E
73B
73E
73B
73E
73B
73E
73B
73E
73B
73C
73C
73C
73C
73D
73D
73D
73C
73C
73C
73D
73D
73D
73C
73D
73D
73D
73D
73D
73D
73D
73D
73D
73D

22-ENKBA-2.0 7000 4
7041 354 00
7042 355 00
7043 356 00
7044 357 01
7045 358 FF
7046 359 FF
7047 360 FF
7048 361 FE
7049 362
7049 363 0C
704A 364 00
704B 365 00
704C 366
704C 367 37
704D 368 F8
704E 369 15
704F 370
704F 371 0100
7051 372 0100
7053 373 FDFF
7055 374 FCFF
7057 375
7057 376 0000
7059 377 8000
705B 378 0000
705D 379 FF7F
705F 380 BFFF
7061 381 FFFF
7063 382 00C0
7065 383
7065 384 AA
7066 385 AA
7067 386 AA
7068 387 AA
7069 388
7069 389 55
706A 390 55
706B 391 55
706C 392 55
706D 393
706D 394 00
706E 395 10
706F 396 01
7070 397
7070 398 0BA4'C3
7073 399 0BA7'C3
7076 400 0BAA'C3
7079 401 0BAD'C3
707C 402
707C 403
707C 404
707C 405
707C 406
707C 407
707C 408
707C 409
707C 410
707C 411 02
707D 412 0000007F
707F 413

N 1
Fiche 1 Frame N1 Sequence 13
ZERO_DAT: .BYTE 0
ONE_SHIFT: .BYTE 0
ONE_DAT: .BYTE 1
ZERO_SHIFT: .BYTE <^XFF>
JMP_INST: .BYTE 0
MOV_CCR_WRO: .BYTE <^X37>
TEST1_DAT: .WORD <^X0100>
TEST2_DAT: .WORD <^X0000>
TEST2_DAT2: .BYTE <^XAA>
TEST2_DAT3: .BYTE <^X55>
TEST7_DAT: .BYTE <^X00>
TEST7_JUMP: JMP TEST7_PAR_ERR
TEMPA_SHIFT: .BYTE 2
TEMPA_DAT: .BLKB 2
STANDARD DATA PATTERNS TO
BE USED BY VARIOUS TESTS.
; LOCATION INTO WHICH JUMP
; INSTRUCTION WILL BE MOVED.
; MOVE INSTRUCTION USED TO PASS
; DATA FROM THE D-BUS LATCH TO
; WRO.
; TEST 1 INPUT DATA.
; TEST 2 INPUT DATA.
; TEST 2 ERROR 4 INPUT DATA.
; TEST 2 ERROR 4 INPUT DATA.
; PARITY CHECKER PATTERN WITH
; BOTH CHECKERS ODD.
; INSTRUCTIONS USED TO RETURN
; TO TEST 7 AFTER A PARITY
; ERROR HAS OCCURRED.
; FIRST TEMPORARY STORAGE LOCATION
; FOR DATA FROM VARIOUS TESTS.

SHIFT VARIABLES - DO NOT SEPARATE PAIRS

22-ENKBA-2.0 7000 4
7041 354 00
7042 355 00
7043 356 00
7044 357 01
7045 358 FF
7046 359 FF
7047 360 FF
7048 361 FE
7049 362
7049 363 0C
704A 364 00
704B 365 00
704C 366
704C 367 37
704D 368 F8
704E 369 15
704F 370
704F 371 0100
7051 372 0100
7053 373 FDFF
7055 374 FCFF
7057 375
7057 376 0000
7059 377 8000
705B 378 0000
705D 379 FF7F
705F 380 BFFF
7061 381 FFFF
7063 382 00C0
7065 383
7065 384 AA
7066 385 AA
7067 386 AA
7068 387 AA
7069 388
7069 389 55
706A 390 55
706B 391 55
706C 392 55
706D 393
706D 394 00
706E 395 10
706F 396 01
7070 397
7070 398 0BA4'C3
7073 399 0BA7'C3
7076 400 0BAA'C3
7079 401 0BAD'C3
707C 402
707C 403
707C 404
707C 405
707C 406
707C 407
707C 408
707C 409
707C 410
707C 411 02
707D 412 0000007F
707F 413

```

22-ENKBA-2.0      7000      4
707F 414 03
7080 415 00000093
7083 416
7083 417 0D
7084 418 00000091
7091 419
7091 420 0D
7092 421 0000009F
709F 422
709F 423 0D
70A0 424 000000AD
70AD 425
70AD 426 04
70AE 427 000000B2
70B2 428
70B2 429 04
70B3 430 000000B7
70B7 431
70B7 432
70B7 433
70B7 434
70B7 435
70B7 436
70B7 437
70B7 438
70B7 439
70B7 440
70B7 441 0000'2A
70BA 442 0008'22
70BD 443
70BD 444 0000'CD
70C0 445
70C0 446 0008'2A
70C3 447 0000'22
70C6 448
70C6 449 C9
70C7 450
70C7 451
70C7 452
70C7 453
70C7 454
70C7 455
70C7 456
70C7 457
70C7 458 40D9 3A
70CA 459 B7
70CB 460 C8
70CC 461 0000'CD
70CF 462
70CF 463 0000'3A
70D2 464 0F
70D3 465 00B7'DC
70D6 466
70D6 467 C9
70D7 468
70D7 469
70D7 470
70D7 471
70D7 472
70D7 473

```

B 2

Fiche 1 Frame B2 Sequence 14

```

TEMPB_SHIFT: .BYTE 3
TEMPB_DAT: .BLKB 3
;SECOND TEMPORARY STORAGE LOCATION
;FOR DATA FROM VARIOUS TESTS.

TEMPC_SHIFT: .BYTE 13
TEMPC_DAT: .BLKB 13
;THIRD TEMPORARY STORAGE LOCATION
;FOR DATA FROM VARIOUS TESTS.

EXP_SHIFT: .BYTE 13
EXP_DAT: .BLKB 13
;LOCATION OF EXPECTED DATA FROM
;VARIOUS TESTS.

REC_SHIFT: .BYTE 13
REC_DAT: .BLKB 13
;LOCATION OF RETURNED DATA FROM
;VARIOUS TESTS.

EXP2_SHIFT: .BYTE 4
EXP2_DAT: .BLKB 4
;LOCATION OF EXPECTED DATA FROM
;TEST 2 ERROR 4.

REC2_SHIFT: .BYTE 4
REC2_DAT: .BLKB 4
;LOCATION OF RETURNED DATA FROM
;TEST 2 ERROR 4.

```

```

*****
;
; CON_SET_FOR_CO THIS ROUTINE IS USED IF A CONTROL C OCCURS TO SAVE THE
; STATE OF THE MACHINE. IF A CONTINUE IS THEN ISSUED, THIS
; ROUTINE RESTORES THE STATE OF THE MACHINE.
;
*****

```

```

CON_SET_FOR_CO: LHL D WCS_ADDRESS ;SAVE THE PRESENT WCS ADDRESS.
                SHLD SAVE_WCS_ADD
                CALL SET_FOR_CONT ;RETURN TO MONITOR.
                LHL D SAVE_WCS_ADD ;RESTORE STATE.
                SHLD WCS_ADDRESS
                RET

```

```

*****
;
; CHECK_RD THIS ROUTINE CHECKS TO SEE IF DIAGNOSTIC IS UNDER RD CONTROL.
; IF IT IS, A CALL IS MADE TO GCVECT SO THAT THE BOOT CODE CAN
; KEEP TRACK OF ELAPSED TIME.
;
*****

```

```

CHECK_RD: LDA MIPFLG ;GET STATE OF MODEM IN PROGRESS
          ORA A ;SET CONDITION CODES
          RZ ;RETURN IF ZERO
          CALL GCVECT ;ELSE GO TO BOOT CODE IF RD
          LDA CNTLC_TYPED ;SEE IF CONTROL C TYPED (UNDER RD)
          RRC
          CC CON_SET_FOR_CO ;IF SO, GO TO MONITOR.
          RET ;BACK TO TEST

```

```

*****
;
; MAKE_NEXT_ADD THIS ROUTINE MOVES THE NEXT ADDRESS FIELD STORED IN THE

```

```

ZZ-ENKBA-2.0      7000      4
70D7 474
70D7 475
70D7 476
70D7 477
70D7 478
70D7 479
70D7 480
70D7 481
70D7 482 C5 46 00'00'21
          77 04 3E
70DF 483
70DF 484 00'7F'21
70E2 485 0000'22
70E5 486 0000'CD
70E8 487
70E8 488 35 00'00'21
          70 C1 00DF'C2
70F1 489
70F1 490 08 3E
70F3 491 00'80'21
70F6 492 B6
70F7 493 0049'32
70FA 494
70FA 495 0081'3A
70FD 496 004A'32
7100 497
7100 498 04 3E
7102 499 00'82'21
7105 500 B6
7106 501 004B'32
7109 502
7109 503 C9
710A 504
710A 505
710A 506
710A 507
710A 508
710A 509
710A 510
710A 511
710A 512
710A 513
710A 514 0112'CD
710D 515
710D 516 23 D3
710F 517 22 D3
7111 518
7111 519 C9
7112 520
7112 521
7112 522
7112 523
7112 524
7112 525
7112 526
7112 527
7112 528
7112 529
7112 530 00'00'11
7115 531 0000'CD

```

C 2

Fiche 1 Frame C2

Sequence 15

LOWER TWO BYTES OF TEMPB_DAT INTO THE NEXT ADDRESS FIELD
OF THE TEST4 JUMP INSTRUCTION AND ALSO ORS IN THE JUMP
INSTRUCTION OP CODE AND JUMP CONTROL CODE INTO THE JUMP
INSTRUCTION.

```

***** MAKE_NEXT_ADD: DO      4
*
*
*
*
*
*****

```

***** MAKE_NEXT_ADD: DO 4

```

LXI    H,TEMPB_SHIFT ;SHIFT THE CONTENTS OF TEMPB_DAT TO
SHLD   SHIFT_PNT     ; THE LEFT FOUR PLACES.
CALL   SHIFTER

```

ENDDO

```

MVI    A,JMP_OP_CODE ;OR THE JUMP INSTRUCTION OP CODE INTO
LXI    H,TEMPB_DAT   ; THE HIGH BYTE OF TEMPB_DAT AND STORE
ORA    M              ; IN THE HIGH BYTE OF JMP_INST.
STA    JMP_INST

```

```

LDA    TEMPB_DAT+1    ;MOVE THE MIDDLE BYTE OF TEMPB_DAT TO
STA    JMP_INST+1     ; THE MIDDLE BYTE OF JMP_INST.

```

```

MVI    A,JMP_CONTROL ;OR THE JUMP CONTROL CODE INTO THE LOW
LXI    H,TEMPB_DAT+2 ; BYTE OF TEMPB_DAT AND STORE IN THE
ORA    M              ; LOW BYTE OF JMP_INST.
STA    JMP_INST+2

```

RET

```

*****
: EXEC_INST THIS ROUTINE EXECUTES AN INSTRUCTION POINTED TO BY THE H,L
: PAIR.
*****

```

```

EXEC_INST: CALL LD_CSR_INST ;LOAD INSTRUCTION INTO THE CSR.

```

```

OUT    SETSS          ;CLOCK THE CPU TO EXECUTE THE
OUT    CLRSS           ; INSTRUCTION.

```

RET

```

*****
: LD_CSR_INST THIS ROUTINE LOADS THE CSR WITH AN INSTRUCTION, WHICH IS
: POINTED TO BY THE CONTENTS OF THE H,L PAIR.
*****

```

```

LD_CSR_INST: LXI    D,CSR_VALUE ;MOVE INSTRUCTION POINTED TO BY THE
CALL    MOVER_3      ; H,L PAIR INTO CSR VALUE.

```


ZZ-ENKBA-2.0	7000	4
7118	532	
7118	533	A2 D3
711A	534	
711A	535	00'00'21
711D	536	0000'22
7120	537	
7120	538	C5 46 00'00'21 77 18 3E
7128	539	
7128	540	38 D3
712A	541	
712A	542	87 DB
712C	543	1F
712D	544	0000'CD
7130	545	
7130	546	0135'D2
7133	547	
7133	548	39 D3
7135	549	
7135	550	25 D3
7137	551	24 D3
7139	552	
7139	553	35 00'00'21 70 C1 0128'C2
7142	554	
7142	555	A3 D3
7144	556	
7144	557	C9
7145	558	
7145	559	
7145	560	
7145	561	
7145	562	
7145	563	
7145	564	
7145	565	
7145	566	
7145	567	
7145	568	
7145	569	
7145	570	0031'2A
7148	571	000C'22
7148	572	
7148	573	00'31'21
714E	574	000E'22
7151	575	
7151	576	C5 46 00'00'21 77 05 3E
7159	577	
7159	578	000E'2A
715C	579	00'00'11
715F	580	02 0E
7161	581	0000'CD
7164	582	
7164	583	00'41'21
7167	584	00'00'11
716A	585	0000'CD
716D	586	
716D	587	0000'CD
7170	588	

A large L-shaped graphic composed of many small stars, forming a corner shape. The stars are arranged in a grid-like pattern, with one arm extending horizontally and the other vertically, meeting at a right angle. The stars are small and black, set against a white background.

15:

```

*****
WCS_SIZER  THIS ROUTINE SIZES THE WCS BY WRITING DATA PATTERNS AT 16K, 17K,
            18K, 19K AND 20K AND THEN READING THESE PATTERNS BACK. IF THE
            PATTERN RECEIVED IS THE SAME AS THE ONE SENT OUT THEN WCS IS
            ASSUMED TO EXIST AT THAT LOCATION. EACH LOACTION IS SIZED TWICE.
*****

```

```

WCS_SIZER:      LHLD      WCS_SIZER_ADDR      ;SET INITIAL WCS SIZE TO 16K.
                  SHLD      WCS_SIZE

                  LXI       H,WCS_SIZER_ADDR    ;SIZE WCS AT 16K.
                  SHLD      WCS_ADDR_PNT

                  DO        5                    ;SIZE AT FIVE LOCATIONS.

                  LHLD      WCS_ADDR_PNT        ;PREPARE TO WRITE TO WCS.
                  LXI       D,WCS_ADDRESS
                  MVI       C,2
                  CALL      MOVER

                  LXI       H,ZERO_DAT          ;LOAD DATA TO BE WRITTEN TO WCS
                  LXI       D,WCS_VALUE         ; INTO WCS_VALUE.
                  CALL      MOVER_3

                  CALL      WCS_WRITE           ;WRITE TO WCS.

```

D 2

Fiche 1 Frame D2 Sequence 16

```

OUT      VSHIFT          ;SET V-BUS TO SHIFTING MODE.

LXI      H,CSR_SHIFT     ;PREPARE TO SHIFT DATA INTO CSR.
SHLD     SHIFT_PNT

DO       24

OUT      CLRCSR           ;CLEAR THE CSR SERIAL INPUT.

IN       CSR23            ;READ THE CSR'S MSB.
RAR      ;ROTATE IT INTO THE CARRY.
CALL     SHIFTER          ;SHIFT THE DATA THROUGH CSR_VALUE.

JNC      1$              ;IF THE CARRY IS SET...

OUT      SETCSR           ;SET THE CSR SERIAL INPUT.

OUT      SETCSK           ;CLOCK THE CSR SERIAL LOAD.
OUT      CLRCSK

ENDDO

OUT      VNORML           ;SET V-BUS TO NORMAL MODE.

RET

*****

THIS ROUTINE SIZES THE WCS BY WRITING DATA PATTERNS AT 16K,
8K, 19K AND 20K AND THEN READING THESE PATTERNS BACK. IF TH
ATTERN RECEIVED IS THE SAME AS THE ONE SENT OUT THEN WCS IS
SSUMED TO EXIST AT THAT LOCATION. EACH LOACTION IS SIZED TW

*****

LHLD     WCS_SIZE_ADDR    ;SET INITIAL WCS SIZE TO 16K.
SHLD     WCS_SIZE

LXI      H,WCS_SIZE_ADDR  ;SIZE WCS AT 16K.
SHLD     WCS_ADDR_PNT

DO       5                ;SIZE AT FIVE LOCATIONS.

LHLD     WCS_ADDR_PNT     ;PREPARE TO WRITE TO WCS.
LXI      D,WCS_ADDRESS
MVI      C,2
CALL     MOVER

LXI      H,ZERO_DATA      ;LOAD DATA TO BE WRITTEN TO WCS
LXI      D,WCS_VALUE      ; INTO WCS_VALUE.
CALL     MOVER_3

CALL     WCS_WRITE        ;WRITE TO WCS.

```

[illegible]

22-ENKBA-2.0	7000	4
71D2	644 00'10'21	
71D5	645 0000'C9	
71D8	646	
71D8	647 00'00'21	
71DB	648 00'00'11	
71DE	649 02 0E	
71E0	650 0000'CD	
71E3	651 02 0E	
71E5	652 0000'CD	
71E8	653	
71E8	654 C9	
71E9	655	
71E9	656	
71E9	657	
71E9	658	
71E9	659	
71E9	660	
71E9	661	
71E9	662	
71E9	663	
71E9	664	
71E9	665	
71E9	666	
71E9	667	
71E9	668 F5	
71EA	669 0000'2A	
71ED	670 4E	
71EE	671 23	
71EF	672 F1	
71F0	673	
71F0	674 7E	
71F1	675 1F	
71F2	676 77	
71F3	677 0D	
71F4	678 C8	
71F5	679 23	
71F6	680	
71F6	681 01F0'C3	
71F9	682	
71F9	683	
71F9	684	
71F9	685	
71F9	686	
71F9	687	
71F9	688	
71F9	689	
71F9	690	
71F9	691	
71F9	692 00'00'11	
71FC	693 05 0E	
71FE	694 0000'CD	
7201	695	
7201	696 C9	
7202	697	
7202	698	
7202	699	
7202	700	
7202	701	
7202	702	
7202	703	

```

F 2
Fiche 1 Frame F2 Sequence 18
WCS_SIZE_P: LXI H,HEADER_B ;PRINT THE WCS SIZE MESSAGE.
CALL PRINT_STRING

LXI H,WCS_SIZE ;PRINT THE HIGHEST WCS ADDRESS.
LXI D,HEX_BUF
MVI C,2
CALL MOVER
MVI C,2
CALL PRINT_HEX

RET

*****
SHIFT_RT THIS ROUTINE SHIFTS N BYTES OF DATA RIGHT 1 BIT. THE LOCATION
POINTED TO BY SHIFT_PNT CONTAINS THE NUMBER OF BYTES TO BE
SHIFTED, THE NEXT N BYTES CONTAINS THE DATA TO BE SHIFTED. THE
CARRY IS SHIFTED INTO THE MSB OF THE SHIFT DATA, AND THE LSB OF
THE SHIFT DATA IS PUT INTO THE CARRY.
*****

SHIFT_RT: PUSH $PSW ;SAVE CARRY.
LHLD SHIFT_PNT ;POINT TO SHIFT DATA COUNT.
MOV C,M ;GET COUNT IN BYTES.
INX H ;POINT TO FIRST BYTE OF SHIFT DATA.
POP $PSW

1$: MOV A,M ;MOVE BYTE 1 RIGHT.
RAR ;CARRY = ROTATE IN BIT NEXT BYTE.
MOV M,A
DCR C ;DEC BYTE COUNT.
RZ ;RETURN IF ZERO.
INX H ;MOVE POINTER TO NEXT BYTE.

JMP 1$

*****
LOAD_MOD_DAT THIS ROUTINE LOADS THE MODULE DATA POINTED TO BY THE H,L
PAIR INTO CON_MOD_DAT.
*****

LOAD_MOD_DAT: LXI D,CON_MOD_DAT ;LOAD THE MODULE DATA POINTED TO BY
MVI C,5 ; THE H,L PAIR INTO CON_MOD_DAT.
CALL MOVER

RET

*****
ERROR_4_1ST THIS ROUTINE IS USED BY TEST1_ERROR AND TEST2_ERROR TO DEFINE
THE ERROR VARIABLES IN PREPARATION FOR USING THE CON_ERROR
ROUTINE.

```

```

ZZ-ENKBA-2.0      7000      4
7202 704
7202 705
7202 706
7202 707 04 0E 00'00'21
          0D 23 77 AF
          0208'C2
720E 708 01 3E
7210 709 0003'32
7213 710
7213 711 0C'92'21
7216 712 00'00'11
7219 713 0000'CD
721C 714
721C 715 00'A0'21
721F 716 00'00'11
7222 717 0000'CD
7225 718
7225 719 04 0E 00'00'21
          0D 23 77 AF
          022B'C2
7231 720
7231 721
7231 722
7231 723 0000'CD
7234 724
7234 725 C9
7235 726
7235 727
7235 728
7235 729
7235 730
7235 731
7235 732
7235 733
7235 734
7235 735
7235 736 02 3E
7237 737 0003'32
723A 738
723A 739 00'96'21
723D 740 00'00'11
7240 741 0000'CD
7243 742
7243 743 00'A4'21
7246 744 00'00'11
7249 745 0000'CD
724C 746
724C 747 0000'CD
724F 748
724F 749 C9
7250 750
7250 751
7250 752
7250 753
7250 754
7250 755
7250 756
7250 757
7250 758
7250 759 0202'CD

```

G 2

Fiche 1 Frame G2

Sequence 19

```

*****
ERROR_4_1ST:  ZERO  CON_ADD_DAT,4  ;STORE A ONE IN THE ADDITIONAL DATA

```

```

MVI  A,1  ; AREA, INDICATING THAT THE FIRST
STA  CON_ADD_DAT+3  ; PART OF THE ERROR MESSAGE FOLLOWS.

```

```

LXI  H,EXP_DAT  ;MOVE FIRST FOUR BYTES OF
LXI  D,CON_EXP_DAT  ;EXPECTED DATA INTO LOCATION
CALL MOVER_4  ;TO BE USED BY ERROR ROUTINE.

```

```

LXI  H,REC_DAT  ;MOVE FIRST FOUR BYTES OF RECEIVED
LXI  D,CON_REC_DAT  ;DATA INTO SIMILAR LOCATION.
CALL MOVER_4

```

```

ZERO  CON_MSK_DAT,4  ;THE FIRST PART OF THE ERROR MASK IS

```

```

; ALL ZEROS, INDICATING THAT ALL BITS
; ARE OF INTEREST.

```

```

CALL  CON_ERROR  ;PRINT FIRST PART OF ERROR MESSAGE.

```

```

RET

```

```

*****
: ERROR_4_2ND  THIS ROUTINE IS USED BY TEST1_ERROR AND TEST2_ERROR TO DEFINE
:               THE ERROR VARIABLES IN PREPARATION FOR USING THE CON_ERROR
:               ROUTINE.
*****

```

```

ERROR_4_2ND:  MVI  A,2  ;SECOND PART OF ERROR MESSAGE.
               STA  CON_ADD_DAT+3

```

```

LXI  H,EXP_DAT+4  ;MOVE SECOND FOUR BYTES OF EXPECTED
LXI  D,CON_EXP_DAT  ;DATA TO LOCATION TO BE USED BY
CALL MOVER_4  ; CON_ERROR ROUTINE.

```

```

LXI  H,REC_DAT+4  ;SAME WITH RECEIVED DATA.
LXI  D,CON_REC_DAT
CALL MOVER_4

```

```

CALL  CON_ERROR  ;PRINT SECOND PART OF ERROR MESSAGE.

```

```

RET

```

```

*****
: TEST1_ERROR  THIS ROUTINE IS USED BY TEST 1 TO DEFINE THE ERROR VARIABLES
:               IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
*****

```

```

TEST1_ERROR:  CALL  ERROR_4_1ST  ;PRINT FIRST PART OF ERROR.

```

```

ZZ-ENKBA-2.0    7000    4
7253 760
7253 761 00'3E
7255 762 0003'32
7258 763
7258 764 0235'CD
725B 765
725B 766 C9
725C 767
725C 768
725C 769
725C 770
725C 771
725C 772
725C 773
725C 774
725C 775
725C 776
725C 777 00'00'21
725F 778 01F9'CD
7262 779
7262 780 0202'CD
7265 781
7265 782 0235'CD
7268 783
7268 784 03 3E
726A 785 0003'32
726D 786
726D 787 00'9A'21
7270 788 00'00'11
7273 789 0000'CD
7276 790
7276 791 00'A8'21
7279 792 00'00'11
727C 793 0000'CD
727F 794
727F 795 0000'CD
7282 796
7282 797 04 3E
7284 798 0003'32
7287 799
7287 800 009E'3A
728A 801 0000'32
728D 802
728D 803
728D 804 00AC'3A
7290 805 0000'32
7293 806
7293 807 FF 3F 21
7296 808 0000'22
7299 809 FFFF 2A
729C 810 0002'22
729F 811
729F 812 0000'CD
72A2 813
72A2 814 C9
72A3 815
72A3 816 00'AE'21
72A6 817 00'00'11
72A9 818 0000'CD
72AC 819

```

H 2

Fiche 1 Frame H2

Sequence 20

```

MVI    A,HEX 3      ;SECOND HALF OF ERROR MASK.
STA    CON_MSK_DAT+3

CALL    ERROR_4_2ND  ;PRINT SECOND PART OF ERROR.

RET

```

```

*****
: TEST2_ERROR THIS ROUTINE IS USED BY TEST 2 TO DEFINE THE ERROR VARIABLES
: IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
*****

```

```

TEST2_ERROR:  LXI    H,MWCS_A      ;WCS MODULE BEING TESTED.
               CALL   LOAD_MOD_DAT

               CALL   ERROR_4_1ST  ;PRINT FIRST QUARTER OF ERROR.

               CALL   ERROR_4_2ND  ;PRINT SECOND QUARTER OF ERROR.

               MVI    A,3          ;THIRD QUARTER OF ERROR MESSAGE.
               STA    CON_ADD_DAT+3

               LXI    H,EXP_DAT+8  ;MOVE THIRD FOUR BYTES OF EXPECTED
               LXI    D,CON_EXP_DAT ; DATA TO LOCATION TO BE USED BY
               CALL   MOVER_4      ; CON_ERROR ROUTINE.

               LXI    H,REC_DAT+8  ;SAME WITH RECEIVED DATA.
               LXI    D,CON_REC_DAT
               CALL   MOVER_4

               CALL   CON_ERROR    ;PRINT THIRD QUARTER OF ERROR MESSAGE.

               MVI    A,4          ;FOURTH QUATER OF ERROR MESSAGE.
               STA    CON_ADD_DAT+3

               LDA    EXP_DAT+12   ;MOVE LAST BYTE OF EXPECTED DATA
               STA    CON_EXP_DAT  ; TO LOCATION TO BE USED BY
                                   ; CON_ERROR ROUTINE.

               LDA    REC_DAT+12   ;SAME WITH RECEIVED DATA.
               STA    CON_REC_DAT

               LXI    H,<^XFF3F>  ;LAST QUARTER OF ERROR MASK OF WHICH
               SHLD   CON_MSK_DAT  ; ONLY THE FIRST TWO BITS ARE OF
               LHLD   <^XFFFF>    ; INTEREST.
               SHLD   CON_MSK_DAT+2

               CALL   CON_ERROR    ;PRINT LAST QUARTER OF ERROR MESSAGE.

               RET

TEST2_ERR4:   LXI    H,EXP2_DAT    ;MOVE EXPECTED DATA TO CON_EXP_DAT.
               LXI    D,CON_EXP_DAT
               CALL   MOVER_4

```

```

22-ENKBA-2.0 7000 4
72AC 820 00'B3'21
72AF 821 00'00'11
72B2 822 0000'CD
72B5 823
72B5 824 0000'32 3C AF
72BA 825
72BA 826 00'00'21
72BD 827 00'00'11
72C0 828 05 0E
72C2 829 0G00'CD
72C5 830
72C5 831 0000'CD
72C8 832 C9
72C9 833
72C9 834
72C9 835
72C9 836
72C9 837
72C9 838
72C9 839
72C9 840
72C9 841
72C9 842
72C9 843 0092'2A
72CC 844 0002'22
72CF 845
72CF 846 00A0'2A
72D2 847 0002'22
72D5 848
72D5 849 FF FF 21
72D8 850 0000'22
72DB 851 00 80 21
72DE 852 0002'22
72E1 853
72E1 854 0000'32 3C AF
72E6 855
72E6 856 00'00'21
72E9 857 01F9'CD
72EC 858
72EC 859 0000'CD
72EF 860
72EF 861 C9
72F0 862
72FC 863
72F0 864
72F0 865
72F0 866
72F0 867
72F0 868
72F0 869
72F0 870
72F0 871
72F0 872
72F0 873 00'92'21
03 0E 00'00'11
0301'CA 0000'CD
72FE 874
72FE 875
72FE 876 0302'CD
7301 877

```

I 2

Fiche 1 Frame 12 Sequence 21

```

LXI H,REC2_DAT ;MOVE RECEIVED DATA TO CON_REC_DAT.
LXI D,CON_REC_DAT
CALL MOVER_4

SET NA_OTHER ;SETUP AND REPORT ERROR.

LXI H,MCPU_A ;CPU MODULE BEING TESTED.
LXI D,CON_MOD_DAT
MVI C,5
CALL MOVER

CALL CON_ERROR ;PRINT AN ERROR
RET

```

```

*****
: TEST3_4_ERROR THIS ROUTINE IS USED BY TESTS 3 AND 4 TO DEFINE THE ERROR
: VARIABLES IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
: *****

```

```

TEST3_4_ERROR: LHLD EXP_DAT ;MOVE EXPECTED DATA TO CON_EXP_DAT.
SHLD CON_EXP_DAT+2

LHLD REC_DAT ;MOVE RECEIVED DATA TO CON_REC_DAT.
SHLD CON_REC_DAT+2

LXI H,<^XFFFF> ;BITS 0-14 ARE OF INTEREST.
SHLD CON_MSK_DAT
LXI H,<^X0080>
SHLD CON_MSK_DAT+2

SET NA_OTHER ;NO ADDITIONAL DATA.

LXI H,MCPU_A ;CPU MODULE BEING TESTED.
CALL LOAD_MOD_DAT

CALL CON_ERROR ; PRINT ERROR MESSAGE.
RET

```

```

*****
: CHECK_5_6_ERROR THIS ROUTINE CHECKS FOR AN ERROR IN TEST 5 AND 6, AND THEN
: CALLS THE ERROR ROUTINE IF AN ERROR WAS DETECTED.
: *****

```

```

CHECK_5_6_ERROR: IFNEQ EXP_DAT,WCS_VALUE,3 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.

CALL TEST5_6_ERROR ;PRINT AN ERROR MESSAGE.

```

22-ENKBA-2.0 7000 4
7301 878
7301 879
7301 880 C9
7302 881
7302 882
7302 883
7302 884
7302 885
7302 886
7302 887
7302 888
7302 889
7302 890
7302 891 00'92'21
7305 892 00'01'11
7308 893 0000'CD
7308 894
7308 895 00'00'21
730E 896 00'01'11
7311 897 0000'CD
7314 898
7314 899 FF 3E
7316 900 0000'32
7319 901 03 0E 00'01'21
0D 23 77 AF
031F'C2
7325 902
7325 903 02 0E 00'00'21
0D 23 77 AF
032B'C2
7331 904 0000'2A
7334 905 0002'22
7337 906
7337 907 00'00'21
733A 908 01F9'CD
733D 909
733D 910 0000'CD
7340 911
7340 912 C9
7341 913
7341 914
7341 915
7341 916
7341 917
7341 918
7341 919
7341 920
7341 921
7341 922
7341 923 FF FF 21
7344 924 0000'22
7347 925 0002'22
734A 926
734A 927 00'80'21
734D 928 00'01'11
7350 929 0000'CD
7353 930
7353 931 00'00'21
7356 932 01F9'CD
7359 933

J 2
ENDIF
RET

Fiche 1 Frame J2

Sequence 22

: TEST5_6_ERROR THIS ROUTINE IS USED BY TESTS 5 AND 6 TO DEFINE THE ERROR
: VARIABLES IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
: *****

TEST5_6_ERROR: LXI H,EXP_DAT ;LOAD 3 BYTES OF EXPECTED DATA
LXI D,CON_EXP_DAT+1 ; INTO CON_EXP_DAT.
CALL MOVER_3

LXI H,WCS_VALUE ;LOAD 3 BYTES OF RECEIVED DATA
LXI D,CON_REC_DAT+1 ; INTO CON_REC_DAT.
CALL MOVER_3

MVI A,<^XFF> ;ERROR MASK INCLUDES LAST THREE
STA CON_MSK_DAT ; BYTES.
ZERO CON_MSK_DAT+1,3

ZERO CON_ADD_DAT,2 ;LOAD THE WCS_POINTER INTO

LHLD WCS_ADDRESS ; CON_ADD_DAT.
SHLD CON_ADD_DAT+2

LXI H,MWCS_A ;WCS MODULE BEING TESTED.
CALL LOAD_MOD_DAT

CALL CON_ERROR ;CALL ERROR ROUTINE.

RET

: TEST7_ERROR THIS ROUTINE IS USED BY TEST 7 TO DEFINE THE ERROR VARIABLES
: IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
: *****

TEST7_ERROR: LXI H,<^XFFFF> ;ERROR MASK IS ALL ONES INDICATING
SHLD CON_MSK_DAT ; THAT NO EXPECTED OR RECEIVED DATA
SHLD CON_MSK_DAT+2 ; IS PRESENT.

LXI H,TEMPB_DAT ;ADDITIONAL DATA CONTAINS DATA SENT
LXI D,CON_ADD_DAT+1 ; TO CSR TO TEST PARITY CIRCUITS.
CALL MOVER_3

LXI H,MCPU_A ;CPU MODULE BEING TESTED.
CALL LOAD_MOD_DAT

```

22-ENKBA-2.0 7000 4
7359 934 0000'32 3C AF
735E 935
735E 936 0000'CD
7361 937
7361 938 C9
7362 939
7362 940
7362 941
7362 942
7362 943
7362 944
7362 945
7362 946
7362 947
7362 948
7362 949 01 3E
7364 950 0000'32
7367 951
7367 952 0092'3A
736A 953 0003'32
736D 954
736D 955 00A0'3A
7370 956 0003'32
7373 957
7373 958 FF FF 21
7376 959 0000'22
7379 960 00 FF 21
737C 961 0002'22
737F 962
737F 963 0000'32 3C AF
7384 964
7384 965 00'00'21
7387 966 01F9'CD
738A 967
738A 968 0000'CD
738D 969
738D 970 C9
738E 971
738E 972
738E 973
738E 974
738E 975
738E 976
738E 977
738E 978
738E 979
738E 980 0092'32
7391 981
7391 982 CC D3
7393 983
7393 984 EC D3
7395 985
7395 986 80 DB
7397 987
7397 988 00A0'32
739A 989
739A 990 00'92'21
01 OF 00'A0'11
03AB'CA 0000'CD
73A8 991

```

```

*****
*
*
*

```

K 2

Fiche 1 Frame K2 Sequence 23

```

SET NA_EXP_REC ;NO EXPECTED AND RECEIVED DATA.
CALL CON_ERROR ;PRINT THE ERROR.
RET

```

```

*****
: TEST8_ERROR THIS ROUTINE IS USED BY TEST 8 TO DEFINE THE ERROR VARIABLES
: IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
*****

```

```

TEST8_ERROR: MVI A,1 ;ERROR NUMBER 1.
STA CON_ERR_NUM

LDA EXP_DAT ;MOVE EXPECTED DATA TO CON_EXP_DAT.
STA CON_EXP_DAT+3

LDA REC_DAT ;MOVE RECEIVED DATA TO CON_REC_DAT.
STA CON_REC_DAT+3

LXI H,<^XFFFF> ;ONLY THE LAST BYTE IS OF
SHLD CON_MSK_DAT ; INTEREST.
LXI H,<^X00FF>
SHLD CON_MSK_DAT+2

SET NA_OTHER ;NO ADDITIONAL DATA.

LXI H,MCPU_A ;CPU MODULE BEING TESTED.
CALL LOAD_MOD_DAT

CALL CON_ERROR ;PRINT THE ERROR MESSAGE.
RET

```

```

*****
: TEST8_1_0 THIS ROUTINE IS USED IN TEST 8 TO LOAD AN 8 BIT DATA PATTERN
: INTO THE ALU AND THEN READ IT BACK
*****

```

```

TEST8_1_0: STA EXP_DAT ; EXP_DAT.
OUT WRITE ;WRITE THIS PATTERN TO THE D-BUS.
OUT YBUSRD ;CLOCK THE Y-BUS LATCH.
IN READ ;READ THE Y-BUS CONTENTS.
STA REC_DAT ;STORE THE Y-BUS CONTENTS IN REC_DAT.
IFNEQ EXP_DAT,REC_DAT,1 ;COMPARE THE EXPECTED AND RECEIVED
; DATA.

```


ZZ-ENKBA-2.0 7000 4
73A8 992
73A8 993 0362'CD
73AB 994
73AB 995
73AB 996
73AB 997 0000'3A
73AE 998 OF
73AF 999 C9
73B0 1000
73B0 1001
73B0 1002
73B0 1003
73B0 1004
73B0 1005
73B0 1006
73B0 1007
73B0 1008
73B0 1009 007D'3A
73B3 1010
73B3 1011 0092'32
73B6 1012
73B6 1013 CC D3
73B8 1014
73B8 1015 EC D3
73BA 1016
73BA 1017 80 DB
73BC 1018
73BC 1019 00A0'32
73BF 1020
73BF 1021 00'92'21
01 0E 00'A0'11
03D0'CA 0000'CD
73CD 1022
73CD 1023
73CD 1024 0362'CD
73D0 1025
73D0 1026
73D0 1027
73D0 1028 0000'3A
73D3 1029 OF
73D4 1030 03B0'DA
73D7 1031 C9
73D8 1032
73D8 1033
73D8 1034
73D8 1035
73D8 1036
73D8 1037
73D8 1038
73D8 1039
73D8 1040
73D8 1041
73D8 1042
73D8 1043
73D8 1044
73D8 1045
73D8 1046
73D8 1047
73D8 1048
73D8 1049

*
*
*

*
*
*
*
*

L 2

Fiche 1 Frame L2

Sequence 24

CALL TEST8_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
RET

: TEST8_SHIFT THIS ROUTINE IS USED BY TEST 8 TO LOAD AN 8 BIT SHIFT PATTERN
: INTO THE ALU AND THEN READ IT BACK
: *****

TEST8_SHIFT: LDA TEMP_DAT ;LOAD THE FOURTH DATA PATTERN INTO
STA EXP_DAT ; A AND INTO EXP_DAT.
OUT WRITE ;WRITE THIS PATTERN TO THE D-BUS.
OUT YBUSRD ;CLOCK THE Y-BUS LATCH.
IN READ ;READ THE Y-BUS CONTENTS.
STA REC_DAT ;STORE THE Y-BUS CONTENTS IN REC_DAT.
IFNEQ EXP_DAT,REC_DAT,1 ;COMPARE THE EXPECTED AND RECEIVED
; DATA.

CALL TEST8_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC ;RETURN WITH C SET IF LOOP ON ERROR
JC TEST8_SHIFT ; AND C CLEAR IF NO LOOP
RET

TEST1 SERIAL SHIFT OF THE UPC

THE UPC WILL BE TESTED BY SHIFTING A ONE THRU A FIELD
OF ZEROS AND THEN A ZERO THRU A FIELD OF ONES. THE DATA WILL BE
SHIFTED THROUGH AS A LARGE STREAM AND COMPARED BIT BY BIT AFTER
THE ENTIRE STRING HAS BEEN SHIFTED THROUGH. THE TEST WILL BE DONE
TWICE. THE FIRST TIME UPC BIT 14 IS READ FROM THE CPU BOARD AND
THE SECOND TIME UPC BIT 14 IS READ FROM THE WCS BOARD.

NOTE: PARITY CHECKING IS OFF DURING THE FOLLOWING TESTS.

```

22-ENKBA-2.0  7000  4
73D8 1050
73D8 1051
73D8 1052
73D8 1053
73D8 1054
73D8 1055
73D8 1056
73D8 1057
73D8 1058
73D8 1059
73D8 1060
73D8 1061
73D8 1062
73D8 1063
73D8 1064
73D8 1065
73D8 1066
73D8 1067
73D8 1068
73D8 1069
73D8 1070
73D8 1071
73D8 1072
73D8 1073
73D8 1074
73D8 1075
73D8 1076
73D8 1077
73D8 1078
73D8 1079
73D8 1080
73D8 1081
73D8 1082
73D8 1083
73D8 1084
73D8 1085
73D8 1086
73D8 1087
73D8 1088
73D8 1089
73D8 1090
73D8 1091
73D8 1092
73D8 1093
73D8 1094
73D8 1095
73D8 1096
73D8 1097
73D8 1098 0000'CD 01 3E
              OF 0000'3A
              C9 03E5'D2
              OF 0000'3A
              3D D3 04E9'DA
73EE 1099
73EE 1100 00'4F'21
73F1 1101 00'92'11
73F4 1102 08 0E
73F6 1103 0000'CD
73F9 1104
73F9 1105 01 3E

```

M 2

Fiche 1 Frame M2

Sequence 25

DATA: 000000000000000100000000000000111111111111101111111111111100

TEST STEPS:

- 1) PUT NOP IN CSR.
- 2) SHIFT IN DATA FROM LIST IN RAM.
- 3) READ OUTPUT DATA FROM CPU BOARD.
- 4) REPEAT 2 + 3 UNTIL ALL DATA IS USED.
- 5) COMPARE INPUT DATA WITH OUTPUT DATA.
- 6) PRINT ERROR IF ANY.
- 7) REPEAT STEPS 1 THROUGH 6 FOR WCS BOARD.

ASSUMPTIONS: ALL CONSOLE RAM TESTS FROM THIS POINT ON ASSUME
THE 8085 HAS BEEN TESTED FULLY. ALSO THAT THE SYSTEM
CLOCK WORKS.

ERROR1: UPC SERIAL SHIFT DATA FAILURE WHEN READ FROM CPU BOARD.
ERROR2: UPC SERIAL SHIFT DATA FAILURE WHEN READ FROM WCS BOARD.

NOTE: BECAUSE OF THE AMOUNT OF DATA SHIFTED THROUGH THE UPC, TWO
ERROR MESSAGES ARE USED TO PRINT THE ENTIRE STRING. ADDITIONAL
DATA SPECIFIES WHICH HALF OF THE DATA STRING IS BEING PRINTED.

DEBUG: 1) THE 'CONS ACK L' SIGNAL SHOULD BE CHECKED BECAUSE IT
IS USED AS THE SERIAL INPUT TO THE UPC.
2) CHECK 'CLK PC H' WHEN SERIAL DATA IS LOADED.
3) CHECK 'PARAL LD CSR H' FOR A LOW VALUE WHEN SERIAL
DATA IS LOADED.
4) CHECK THE SERIAL INTERCONNECT BETWEEN CHIPS.
5) CHECK 'UPC SHF OUT H' FOR SERIAL DATA COMING OUT.
6) CHECK WCS BITS 8,9,10 FOR 0 SELECTING THE 'UPC SHF OUT H'
SIGNAL TO 'BUS AD7 H'.
7) CHECK 'SEL STATUS L' FOR ENABLING THE STATUS
SELECT MUX.

TEST1: HEAD <^X01> ;MACRO FOR INITIALIZING TEST 1.

```

LXI H,TEST1_DAT ;COPY THE TEST DATA INTO THE
LXI D,EXP_DAT ; EXP_DAT AREA.
MVI C,8
CALL MOVER
MVI A,1 ;FIRST PART OF TEST 1.

```


22-ENKBA-2.0 7000 4

```

7469 1162 *
7469 1163 *****
7469 1164
7469 1165 0000'3A
746C 1166 0F
746D 1167 0404'DA
7470 1168
7470 1169 02 3E
7472 1170 0000'32
7475 1171
7475 1172 00'00'21
7478 1173 01F9'CD
747B 1174
747B 1175 0000'CD
747E 1176
747E 1177 00'4F'21
7481 1178 00'84'11
7484 1179 08 0E
7486 1180 0000'CD
7489 1181
7489 1182 A2 D3
748B 1183
748B 1184 C5 46 00'00'21
77 4E 3E
7493 1185
7493 1186
7493 1187
7493 1188 00'83'21
7496 1189 0000'22
7499 1190 0C00'CD
749C 1191
749C 1192 04A4'D2
749F 1193
749F 1194 A9 D3
74A1 1195
74A1 1196 04A6'C3
74A4 1197
74A4 1198 A8 D3
74A6 1199
74A6 1200
74A6 1201
74A6 1202 27 D3
74A8 1203 26 D3
74AA 1204
74AA 1205 00'9F'21
74AD 1206 0000'22
74B0 1207 00 DB
74B2 1208 17
74B3 1209
74B3 1210 0000'CD
74B6 1211
74B6 1212 00C7'CD
74B9 1213
74B9 1214 35 00'00'21
70 C1 0493'C2
74C2 1215
74C2 1216 00'A5'21
74C5 1217 00'A0'11
74C8 1218 08 0E
74CA 1219 0000'CD

```

2\$:

B 3

Fiche 1 Frame B3

Sequence 27

```

ENDIF
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 1$ ;...JUMP TO BEGINNING OF ERROR LOOP.
MVI A,2 ;SECOND PART OF TEST 1.
STA CON_ERR_NUM
LXI H,MWCS_A ;WCS MODULE BEING TESTED.
CALL LOAD_MOD_DAT
CALL NOP_CSR ;KEEP CPU IDLE.
LXI H,TEST1_DAT ;COPY THE 8 BYTE TEST DATA STRING
LXI D,TEMPC_DAT ; INTO THE TEMPC_DAT AREA.
MVI C,8
CALL MOVER
OUT VSHIFT ;PUT CPU IN SHIFTING MODE.
DO 78 ;78 SHIFTS MUST BE MADE TO PASS
;THE ENTIRE 64 BIT DATA STRING
;THROUGH THE 15 BIT UPC REGISTER.
LXI H,TEMPC_SHIFT ;SEND A BIT OF DATA TO BE SHIFTED
SHLD SHIFT_PNT ;THROUGH THE UPC.
CALL SHIFTER
IFC
OUT SETUPC ;SET TO SHIFT IN A ONE INTO THE UPC.
ELSE
OUT CLRUPC ;CLEAR TO SHIFT IN A ZERO INTO THE UPC.
ENDIF
OUT SETUPK ;CLOCK THE UPC REGISTER.
OUT CLRUPK
LXI H,REC_SHIFT ;BRING IN A BIT FROM THE UPC
SHLD SHIFT_PNT ; AND SHIFT IT INTO THE REC_DAT
IN UPC14A ; AREA.
RAL
CALL SHIFTER
CALL CHECK_RD ;CALL BOOT CODE IF UNDER RD CONTROL
ENDDO
LXI H,REC_DAT+5 ;MOVE THE RETURNED DATA FROM THE
LXI D,REC_DAT ; BOTTOM TO THE TOP OF THE REC_DAT
MVI C,8 ; AREA.
CALL MOVER

```

ZZ-ENKBA-2.0 7000 4
74CD 1220
74CD 1221 A3 D3
74CF 1222
74CF 1223 00'92'21
08 0E 00'A0'11
04E0'CA 0000'CD
74DD 1224
74DD 1225
74DD 1226 0250'CD
74E0 1227
74E0 1228
74E0 1229
74E0 1230 0000'3A
74E3 1231 0F
74E4 1232 047B'DA
74E7 1233
74E7 1234 3C D3
74E9 1235
74E9 1236
74E9 1237
74E9 1238
74E9 1239
74E9 1240
74E9 1241
74E9 1242
74E9 1243
74E9 1244
74E9 1245
74E9 1246
74E9 1247
74E9 1248
74E9 1249
74E9 1250
74E9 1251
74E9 1252
74E9 1253
74E9 1254
74E9 1255
74E9 1256
74E9 1257
74E9 1258
74E9 1259
74E9 1260
74E9 1261
74E9 1262
74E9 1263
74E9 1264
74E9 1265
74E9 1266
74E9 1267
74E9 1268
74E9 1269
74E9 1270
74E9 1271
74E9 1272
74E9 1273
74E9 1274
74E9 1275
74E9 1276
74E9 1277

*
*
*
*

C 3

Fiche 1 Frame C3

Sequence 28

OUT VNORML ;RETURN CPU TO NORMAL MODE.
IFNEQ EXP_DAT,REC_DAT,8 ;COMPARE THE EXPECTED AND
; RECEIVED DATA.
CALL TEST1_ERROR ;PRINT AN ERROR MESSAGE.
ENDIF
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 2\$;...JUMP TO BEGINNING OF ERROR LOOP.
TAIL ;MACRO FOR ENDING TEST.

TEST2

SERIAL SHIFT OF THE CSR

THE CSR WILL BE TESTED BY SHIFTING THE FOLLOWING DATA
THROUGH THE REGISTER. THE RESULTS WILL FIRST BE CHECKED AT BIT 7
THEN AT BIT 15 AND AGAIN AT BIT 23. A SIMILAR PROCEDURE IS THEN
USED TO CHECK THE SELECT LINES TO THE LOAD CSR LATCH

DATA: 0000000000000000000000000100000000000000000000000
11111111111111111111111101111111111111111111111111111

TEST STEPS:

- 1) SHIFT IN DATA FROM LIST IN RAM.
- 2) READ OUTPUT DATA AT BIT 7.
- 3) REPEAT 1 + 2 UNTIL ALL DATA IS USED.
- 4) COMPARE INPUT DATA WITH OUTPUT DATA.
- 5) PRINT ERROR IF ANY.
- 6) REPEAT 1 - 5 READING OUTPUT DATA AT BIT 15.
- 7) REPEAT 1 - 5 READING OUTPUT DATA AT BIT 23.
- 8) LOAD CSR WITH TEST DATA
- 9) SET UP TO LOAD CSR WITH COMPLEMENT OF DATA, BUT
DO NOT PUT CPU INTO SHIFT MODE
- 10) EXAMINE CSR.
- 11) IF CSR HAS NEW DATA, PRINT ERROR

ASSUMPTIONS: BUS AD FROM THE 8085 IS WORKING ON BIT 7 AND
THAT THE ENABLE OF THE CLOCK CSR SIGNAL WORKS.

ERROR1: CSR SERIAL SHIFT DATA FAILURE AT CSR BIT 7.
ERROR2: CSR SERIAL SHIFT DATA FAILURE AT CSR BIT 15.
ERROR3: CSR SERIAL SHIFT DATA FAILURE AT CSR BIT 23.
ERROR4: CSR LOADED DATA WHEN CPU WAS NOT IN SHIFT MODE.

```

74E9 1278
74E9 1279
74E9 1280
74E9 1281
74E9 1282
74E9 1283
74E9 1284
74E9 1285
74E9 1286
74E9 1287
74E9 1288
74E9 1289
74E9 1290
74E9 1291
74E9 1292
74E9 1293
74E9 1294
74E9 1295
74E9 1296
74E9 1297
74E9 1298
74E9 1299

```

```

0000'CD 02 3E
OF 0000'3A
C9 04F6'D2
OF 0000'3A
3D D3 06FB'DA

```

```

74FF 1300
74FF 1301 01 3E
7501 1302 0000'32
7504 1303
7504 1304 00'57'21
7507 1305 00'92'11
750A 1306 0D 0E
750C 1307 0000'CD
750F 1308
750F 1309 00'57'21
7512 1310 00'84'11
7515 1311 0D 0E
7517 1312 0000'CD
751A 1313
751A 1314 A2 D3
751C 1315
751C 1316 C5 46 00'00'21
77 6F 3E

```

```

7524 1317
7524 1318
7524 1319
7524 1320
7524 1321 00'83'21
7527 1322 0000'22
752A 1323 0000'CD
752D 1324
752D 1325 0535'D2
7530 1326
7530 1327 39 D3
7532 1328
7532 1329 0537'C3
7535 1330
7535 1331 38 D3
7537 1332

```

NOTE: ADDITIONAL DATA WILL SPECIFY WHICH QUARTER OF THE ERROR MESSAGE IS BEING PRINTED.

DEBUG: 1) CHECK 'WCS WRT DATA 7 H' FOR DATA BEING SERIALY SHIFTED INTO THE CSR.
2) CHECK 'WRT WWD REG H' TO SEE THAT 'WCS WRT DATA 7 H' IS BEING CLOKED INTO THE REGISTER.
3) CHECK 'PARAL LD CRS H' FOR LOW VALUE WHEN A SERIAL SHIFT IS BEING USED.
4) CHECK '+3VC H' ON THE CLR INPUT TO THE SHIFT REG.
5) CHECK 'CLK CSR H' WHEN DATA IS BEING SHIFTED.
6) CHECK THE CONECTIONS OF SIGNALS BETWEEN EACH 4 BITS OF THE CSR FOR THAT DATA TO SHIFT TO THE NEXT STAGE.
ERROR4: CHECK SELECT LINE 2 TO PARAL LD CSR LATCH FOR STUCK LOW.

TEST2: HEAD <^X02>

1\$:

```

MVI A,1 ;FIRST PART OF TEST 2.
STA CON_ERR_NUM

LXI H,TEST2_DAT ;COPY THE TEST 2 DATA INTO
LXI D,EXP_DAT ; INTO THE EXP_DAT AREA.
MVI C,13
CALL MOVER

LXI H,TEST2_DAT ;COPY 13 BYTES OF TEST DATA
LXI D,TEMPC_DAT ; INTO TEMPC_DAT.
MVI C,13
CALL MOVER

OUT VSHIFT ;PUT CPU IN SHIFTING MODE.

DO 111 ;111 SHIFTS MUST BE MADE TO

; PASS THE ENTIRE 104 DATA BITS
; THROUGH THE LOW EIGHT BITS OF
; THE CSR.

LXI H,TEMPC_SHIFT ;SHIFT A BIT INTO THE CSR BY EITHER
SHLD SHIFT_PNT ; SETTING OR CLEARING THE CSR SERIAL
CALL SHIFTER ; INPUT AND CLOCKING CSK.

IFC

OUT SETCSR

ELSE

OUT CLRCR

```

Station	Altitude	Latitude	Longitude	Remarks
7537	1333			
7537	1334			
7537	1335	25	D3	
7539	1336	24	D3	
7538	1337			
7538	1338	00'9F'21		
753E	1339	0000'22		
7541	1340	85	DB	
7543	1341	1F		
7544	1342	0G00'CD		
7547	1343			
7547	1344	00C7'CD		
754A	1345			
754A	1346	35 00'00'21		
		70 C1 0524'C2		
7553	1347			
7553	1348	A3	D3	
7555	1349			
7555	1350	00'92'21		
		0D 0E 00'A0'11		
		0566'CA 0000'CD		
7563	1351			
7563	1352			
7563	1353	025C'CD		
7566	1354			
7566	1355			
7566	1356			
7566	1357	0000'3A		
7569	1358	0F		
756A	1359	050F'DA		
756D	1360			
756D	1361			
756D	1362	02 3E		
756F	1363	0000'32		
7572	1364			
7572	1365	00'57'21		
7575	1366	00'84'11		
7578	1367	0D 0E		
757A	1368	0000'CD		
757D	1369			
757D	1370	A2	D3	
757F	1371			
757F	1372	C5 46 00'00'21		
		77 77 3E		
7587	1373			
7587	1374			
7587	1375	00'83'21		
758A	1376	0000'22		
758D	1377	0000'CD		
7590	1378			
7590	1379	0598'D2		
7593	1380			
7593	1381	39	D3	
7595	1382			
7595	1383	059A'C3		
7598	1384			
7598	1385	38	D3	
759A	1386			
759A	1387			
759A	1388			

E 3

Fiche 1 Frame E3

Sequence 30

ENDIF

```
OUT      SETCSK
OUT      CLRC SK
```

```

LXI    H,REC_SHIFT      ;SHIFT A BIT FROM THE CSR INTO THE
SHLD   SHIFT_PNT        ; RETURN DATA AREA
IN     CSR07

```

CALL SHIFTER

```
CALL    CHECK_RD      ;CALL BOOT CODE IF UNDER RD CONTROL
```

ENDDO

```
OUT    VNORML      ;RETURN THE CPU TO NORMAL MODE.
```

```
IFNEQ EXP_DAT,REC_DAT,13 ;COMPARE THE EXPECTED AND
```

```

; RECEIVED DATA.

```

```
CALL TEST2_ERROR ;PRINT ERROR MESSAGE.
```

ENDIF

```
LDA    ERROR_CON    ;IF THE ERROR LOOPING FLAG IS SET...
```

```

JC      18      ;...JUMP TO BEGINNING OF ERROR LOOP.

```

```

MVI      A,2                ;SECOND PART OF TEST 2.
STA      CON_ERR_NUM

```

```

LXI      H,TEST2_DAT      ;COPY 13 BYTES OF TEST DATA INTO
LXI      D,TEMPC_DAT      ; TEMPC_DAT.
MVI      C,13
CALL     MOVER

```

OUT VSHIFT ;PUT CPU IN SHIFT MODE.

DO 119 ;THE LOW 16 BITS OF THE CSR ARE

; NOW BEING TESTED.

```
LXI    H,TEMPC_SHIFT    ;SHIFT A BIT OF DATA FROM THE TEMP
SHLD   SHIFT_PNT        ; DATA AREA INTO THE WCS REGISTER...
CALL   SHIFTER
```

IFC

OUT SETCSR

ELSE

OUT CLRCR

ENDIF

[illegible]

22-ENKBA-2.0	7000	4
759A 1389	25 D3	
759C 1390	24 D3	
759E 1391		
759E 1392	00'9F'21	
75A1 1393	0000'22	
75A4 1394	86 DB	
75A6 1395	1F	
75A7 1396	0000'CD	
75AA 1397		
75AA 1398	00C7'CD	
75AD 1399		
75AD 1400	35 00'00'21	
	70 C1 0587'C2	
75B6 1401		
75B6 1402	A3 D3	
75B8 1403		
75B8 1404	00'92'21	
	0D 0E 00'AO'11	
	05C9'CA 0000'CD	
75C6 1405		
75C6 1406		
75C6 1407	025C'CD	
75C9 1408		
75C9 1409		
75C9 1410		
75C9 1411	0000'3A	
75CC 1412	0F	
75CD 1413	0572'DA	
75D0 1414		
75D0 1415	03 3E	
75D2 1416	0000'32	
75D5 1417		
75D5 1418	00'57'21	
75D8 1419	00'84'11	
75DB 1420	0D 0E	
75DD 1421	0000'CD	
75E0 1422		
75E0 1423	A2 D3	
75E2 1424		
75E2 1425	C5 46 00'00'21	
	77 7F 3E	
75EA 1426		
75EA 1427		
75EA 1428	00'83'21	
75ED 1429	0000'22	
75F0 1430	0000'CD	
75F3 1431		
75F3 1432	05FB'D2	
75F6 1433		
75F6 1434	39 D3	
75F8 1435		
75F8 1436	05FD'C3	
75FB 1437		
75FB 1438	38 D3	
75FD 1439		
75FD 1440		
75FD 1441		
75FD 1442	25 D3	
75FF 1443	24 D3	
7601 1444		

F 3

Fiche 1 Frame F3 Sequence 31
;...AND THEN INTO THE CSR.

[illegible]

ZZ-ENKBA-2.0	7000	4
7601 1445	00'9F'21	
7604 1446	0000'22	
7607 1447	87 DB	
7609 1448	1F	
760A 1449	0000'CD	
760D 1450		
760D 1451	00C7'CD	
7610 1452		
7610 1453	35 00'00'21	
	7C C1 05EA'C2	
7619 1454		
7619 1455	A3 D3	
761B 1456		
761B 1457	00'92'21	
	0D 0E 00'AO'11	
	062C'CA 0000'CD	
7629 1458		
7629 1459		
7629 1460	025C'CD	
762C 1461		
762C 1462		
762C 1463		
762C 1464	0000'3A	
762F 1465	0F	
7630 1466	05D5'DA	
7633 1467		
7633 1468		
7633 1469	04 3E	
7635 1470	0000'32	
7638 1471		
7638 1472	00 FF 21	
763B 1473	0000'22	
763E 1474	03 0E 00'01'21	
	0D 23 77 AF	
	0644'C2	
764A 1475		
764A 1476	00'65'21	
764D 1477	0112'CD	
7650 1478		
7650 1479	00'69'21	
7653 1480	00'AF'11	
7656 1481	0000'CD	
7659 1482		
7659 1483	04 0E 00'B3'21	
	0D 23 77 AF	
	065F'C2	
7665 1484		
7665 1485	00'69'21	
7668 1486	00'00'11	
766B 1487	0000'CD	
766E 1488		
766E 1489	A6 D3	
7670 1490		
7670 1491		
7670 1492		
7670 1493	00'00'21	
7673 1494	0000'22	
7676 1495		
7676 1496	00C7'CD	
7679 1497		

G 3

Fiche 1 Frame G3 Sequence 32

```
CALL    CHECK_RD          ;CALL BOOT CODE IF UNDER RD CONTROL
ENDDO
```

```
OUT      VNORML      :PUT CPU IN NORMAL MODE.
```

```
IFNEQ EXP_DAT,REC_DAT,13      ;COMPARE THE EXPECTED AND
```

: RECEIVED DATA.

```
CALL TEST2_ERROR ;PRINT AN ERROR MESSAGE.
```

ENDIF

```
LDA    ERROR_CON    ;IF THE ERROR LOOPING FLAG IS SET...
```

```

JC      38      ;...JUMP TO BEGINNING OF ERROR LOOP.

```

MVI A,4 ;FOURTH PART OF TEST 2

STA	CON_ERR_NUM
------------	--------------------

LXI H,<XFF> ;HIGHEST BYTE IS MASKED

```
SHLD CON_MSK_DAT
ZERO CON_MSK_DAT+1,3
```

```
LXI    H,TEST2_DAT2      ;LOAD CSR WITH DATA OF
CALL    LD_CSR_INST       ; ALTERNATING 1'S AND 0'S.
```

```

LXI    H,TEST2_DAT3      ;COMPLEMENT OF DATA PATTERN
LXI    D,EXP2_DAT+1      ; MOVED INTO EXP_DAT
CALL   MOVER_3

```

ZERO REC2_DAT,4 ;ZERO RECEIVED DATA AREA

```

LXI    H,TEST2_DAT3    ;MOVE TEST DATA INTO CSR_VALUE

```

```

LXI    D, CSR_VALUE
CALL   MOVER-3

```

```
OUT      ENBMEM      ;ENABLE MEM REF
                        ;THIS WILL SET V-BUS TO SHIFTING MODE
                        ; IF SELECT LINE 2 STUCK LOW
```

```
LXI    H,CSR_SHIFT    ;SET UP TO SHIFT DATA INTO CSR
```

```
CALL    CHECK_RD      ;CALL BOOT CODE IF UNDER RD CONTROL
```


Sequence 34

```

OUT      VNORMAL      ;PUT CPU BACK IN NORMAL MODE

IFEQ     EXP2_DAT+1,REC2_DAT+1,3 ;SECOND DATA PATTERN SHOULD NOT

                                ; BE IN CSR.

CALL     TEST2_ERR4     ;PRINT AN ERROR MESSAGE

ENDIF

LDA      ERROR_CON      ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC       4$              ;...JUMP TO BEGINNING OF ERROR LOOP.

TAIL     ;MACRO FOR ENDING TEST.

```

TEST3 UPC NEXT ADDRESS TEST

THE UPC WILL BE LOADED WITH ADDRESSES THAT WILL CAUSE CARRIES TO THE NEXT HIGHER BITS AND THE CSR WILL EXECUTE A NOP.

UPC SENT	RETURNED
000000000000001	000000000000010
0000000000000011	0000000000000100
0000000000000111	0000000000001000
0000000000001111	0000000000010000
0000000000011111	0000000000100000
0000000001111111	0000000010000000
0000000011111111	0000000100000000
0000000111111111	0000001000000000
0000001111111111	0000010000000000
0000001111111111	0000100000000000
0000011111111111	0001000000000000
0000111111111111	0010000000000000
0001111111111111	0010000000000000
0011111111111111	0100000000000000
0111111111111111	1000000000000000

TEST STEPS:

- 1) LOAD UPC WITH ADDRESS THAT WILL CAUSE A CARRY.
- 2) INCREMENT THAT PATTERN AND PUT IT IN EXP_DAT.
- 2) LOAD CSR WITH NOP AND EXECUTE.
- 3) READ AND COMPARE UPC TO EXPECTED DATA.
- 4) PRINT ERRORS IF ANY.
- 5) REPEAT 1 - 4 FOR ALL APPROPRIATE PATTERNS.

ASSUMPTIONS: THE SERIAL LOAD AND READ OF THE UPC IS WORKING AS TESTED IN PREVIOUS TESTS.

[illegible]

```

ZZ-ENKBA-2.0      7000      4
76FB 1612
76FB 1613
76FB 1614
76FB 1615
76FB 1616
76FB 1617
76FB 1618
76FB 1619
76FB 1620
76FB 1621
76FB 1622
76FB 1623
76FB 1624
76FB 1625
76FB 1626
76FB 1627
76FB 1628
76FB 1629
76FB 1630
76FB 1631
76FB 1632 0000'CD 03 3E
              OF 0000'3A
              C9 0708'D2
              OF 0000'3A
              3D D3 0779'DA
7711 1633
7711 1634 01 3E
7713 1635 0000'32
7716 1636
7716 1637 0043'2A
7719 1638 007D'22
771C 1639
771C 1640 C5 46 00'00'21
              77 0E 3E
7724 1641
7724 1642
7724 1643 007D'2A
7727 1644 0092'22
772A 1645
772A 1646 00'92'21
772D 1647 0000'CD
7730 1648
7730 1649 007D'2A
7733 1650 0000'22
7736 1651
7736 1652 0000'CD
7739 1653
7739 1654 0000'CD
773C 1655
773C 1656 23 D3
773E 1657 22 D3
7740 1658
7740 1659
7740 1660 0000'CD
7743 1661
7743 1662 0000'2A
7746 1663 00A0'22
7749 1664
7749 1665 00'92'21
              02 0E 00'A0'11

```

J 3

Fiche 1 Frame J3

Sequence 35

ERROR1: UPC NEXT ADDRESS FAILURE.

DEBUG: 1) CHECK 'BUS PC H' SIGNALS TO SEE THAT THE ADDRESS IS COMING OUT OF THE UPC.
 2) CHECK 'EN PC L' TO ENABLE THE OUTPUTS.
 3) CHECK THAT THE STACK RAMS ARE NOT HOLDING 'BUS PC H' SIGNALS LOW.
 4) CHECK THAT 'ENABLE JUMP L' IS HIGH ON THE INCR. -HI PAL.
 5) CHECK THAT 'LOW INC COUT H' IS LOW ON THE INCR. -HI PAL.
 6) CHECK THAT 'JUMP INSTR L' IS HIGH ON MUX AND INCR CTL COMING FROM THE INSTRUCTION DECODE LOGIC.
 7) CHECK THAT 'INCR CIN H' IS LOW.
 8) CHECK THAT 'BUS NAD H' IS NOT SHORTED OR HELD LOW.
 9) CHECK THAT 'ENABLE JUMP LO L' IS HIGH ON CSR.OS MUX PAL.

TEST3: HEAD <^X03>

1\$:

```

MVI A,1 ;FIRST (AND ONLY) PART OF TEST 3.
STA CON_ERR_NUM

LHLD ONE_SHIFT+1 ;LOAD THE FIRST DATA PATTERN TO BE
SHLD TEMPA_DAT ; SENT TO THE UPC INTO TEMPA_DAT.

DO 14 ;14 DATA WORDS WILL BE LOADED INTO
      ; THE UPC.

LHLD TEMPA_DAT ;MOVE A UPC ADDRESS TO EXP_DAT...
SHLD EXP_DAT

LXI H,EXP_DAT ;...AND INCREMENT IT.
CALL INC_WORD

LHLD TEMPA_DAT ;MAOV A UPC ADDRESS INTO UPC_VALUE.
SHLD UPC_VALUE

CALL LOAD_UPC ;LOAD THE UPC FROM UPC_VALUE.

CALL NOP_CSR ;LOAD A NOP INSTRUCTION INTO THE CPU.

OUT SETSS ;SET AND CLEAR THE SINGLE STEP TO
OUT CLRSS ; EXECUTE ONE INSTRUCTION, THUS
           ; INCREMENTING THE UPC.

CALL LOAD_UPC ;LOAD THE UPC INTO UPC_VALUE.

LHLD UPC_VALUE ;MOVE THE DATA RETURNED FROM THE
SHLD REC_DAT ; UPC TO REC_DAT.

IFNEQ EXP_DAT,REC_DAT,2 ;COMPARE THE EXPECTED AND

```


77DB 1776 0600'2A

[illegible]

15:

Fiche 1 Frame L3 Sequence 37

ARE BOTH LOW FORM THE SEQ.CTL PAL.

- 2) CHECK FOR 'ENABLE JUMP LO L' ON CSR.OS MUX PAL AND ON THE MUX SELECTING THE CSR BITS.
- 3) CHECK BUS NAD FROM THE CSR.OS MUX PAL (BITS 0 - 4).
- 4) CHECK THAT THE CSR BITS ARE SELECTED TO NAD 4 - 13 AND THAT WCS PAGE IS SELECTED TO NAD 14 ON THE MUX.
- 5) CHECK THAT THE WCS PAGE BIT IS CORRECT FOR THE STATE REG MISC DECODER PAL.

```

HEAD      <^X04>                ;MACRO FOR INITIALIZING TEST 4.

MVI      A,1                      ;FIRST PART OF TEST 4.
STA      CON_ERR_NUM

LXI      H,X_CLR_PAGE+1          ;CLEAR THE WCS PAGE BIT LOADER.
CALL     EXEC_INST

LHLD     ONE_SHIFT+1              ;LOAD FIRST NEXT ADDRESS FIELD
SHLD     TEMPB_DAT                ; INTO TEMPB_DAT.

CALL     CHECK_RD                 ;CALL BOOT CODE IF UNDER RD CONTROL

DO        14                      ;MAIN LOOP FOR EXECUTING THE JUMP
                                ; INSTRUCTION FOR EACH OF THE 14 NEXT
                                ; ADDRESSES.

ZERO     TEMPB_DAT,3              ;CLEAR TEMPB_DAT.

LHLD     TEMPB_DAT                ;COPY THE ADDRESS IN TEMPB_DAT INTO
SHLD     EXP_DAT                  ; EXP DAT AND INTO THE LOWER TWO
SHLD     TEMPB_DAT+1              ; BYTES OF TEMPB_DAT.

LXI      H,EXP_DAT                ;INCREMENT EXP DAT AS THE CSR NEXT
CALL     INC_WORD                 ; ADDRESS FIELD IS INCREMENTED WHEN
                                ; IT IS SENT TO THE UPC.

CALL     MAKE_NEXT_ADD            ;OR IN THE ADDRESS STORED IN TEMPB_DAT
                                ; INTO THE JUMP INSTRUCTION.

LXI      H,0                      ;CLEAR THE UPC.
SHLD     UPC_VALUE
CALL     LOAD_UPC

LXI      H,JMP_INST              ;EXECUTE THE JUMP INSTRUCTION.
CALL     EXEC_INST

CALL     LOAD_UPC                 ;MOVE THE UPC CONTENTS INTO UPC_VALUE.

LHLD     UPC_VALUE                ;MOVE THE VALUE RETURNED FROM THE UPC

```

[illegible]

ZZ-ENKBA-2.0 7000 4
02 OE 00'AO'11
085B'CA 0000'CD

7858 1832
7858 1833
7858 1834 02C9'CD
7858 1835
7858 1836
7858 1837
7858 1838 0000'3A
785E 1839 0F
785F 1840 082C'DA
7862 1841
7862 1842
7862 1843 3C D3
7864 1844
7864 1845
7864 1846
7864 1847
7864 1848
7864 1849
7864 1850
7864 1851
7864 1852
7864 1853
7864 1854
7864 1855
7864 1856
7864 1857
7864 1858
7864 1859
7864 1860
7864 1861
7864 1862
7864 1863
7864 1864
7864 1865
7864 1866
7864 1867
7864 1868
7864 1869
7864 1870
7864 1871
7864 1872
7864 1873
7864 1874
7864 1875
7864 1876
7864 1877
7864 1878
7864 1879
7864 1880
7864 1881
7864 1882
7864 1883
7864 1884
7864 1885
7864 1886
7864 1887
7864 1888
7864 1889

*
*
*
*
*
*

N 3

Fiche 1 Frame N3

Sequence 39

; RECEIVED DATA.

CALL TEST3_4_ERROR ;PRINT AN ERROR MESSAGE.

ENDIF

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...

RRC
JC 2\$;...JUMP TO BEGINNING OF ERROR LOOP.

TAIL

TEST5

WCS RAM DATA TEST

THE WCS RAM DATA TESTS LOCATION ZERO OF THE RAM WITH
A PATTERN OF ALL 1'S, ALL 0'S, A ZERO RIPPLED THROUGH A FIELD
OF ONES AND A ONE RIPPLED THROUGH A FIELD OF ZERO'S. ALL OTHER
LOCATIONS WILL BE CHECKED WITH ALL 1'S AND ALL 0'S ONLY. WCS
WILL BE SIZED BY WRITING INTO HIGH ADDRESSES AND THEN READING AT
THESE ADDRESS TO SEE IF THE DATA CAN BE WRITTEN TO THESE ADDRESS.
ALL WCS LOCATIONS ABOVE 16K WILL BE TESTED BY SHIFTING 4 BIT BLOCKS
OF A ONE THROUGH A ZERO FIELD AND A ZERO THROUGH A ONE FIELD AT
EACH LOCATION. ALL WCS FOUND WILL BE TESTED.

TEST STEPS:

- 1) SET ADDRESS 0 IN UPC.
- 2) WRITE SHITING DATA PATTERN.
- 3) READ SHIFTING DATA PATTERN.
- 4) COMPARE SENT DATA TO RETURNED DATA.
- 5) PRINT ERROR IF ANY.
- 6) REPEAT 2 - 5 FOR EACH DATA PATTERN AT ADDRESS 0.
- 7) WRITE ALL 0'S, READ AND COMPARE TO ALL 0'S.
- 8) PRINT ERROR IF ANY.
- 9) WRITE ALL 1'S, READ AND COMPARE TO ALL 1'S.
- 10) PRINT ERROR IF ANY.
- 11) REPEAT 7 - 10 FOR EACH MEMORY LOCATION BELOW 16K.
- 12) WRITE FOUR BIT BLOCK PATTERN.
- 13) READ FOUR BIT BLOCK PATTERN.
- 14) COMPARE SENT DATA TO RETURNED DATA.
- 15) PRINT ERROR IF ANY.
- 16) REPEAT 12-15 FOR EACH PATTERN.
- 17) REPEAT 12-16 FOR EACH LOACTION ABOVE 16K.

ASSUMPTIONS: THE UPC AND CSR ARE WORKING AS TESTED
IN PREVIOUS TESTS.

ERROR1: WCS RAM DATA TEST FAILURE AT LOCATION ZERO.

ZZ-ENKBA-2.0 7000 4

7864 1890
7864 1891
7864 1892
7864 1893
7864 1894
7864 1895
7864 1896
7864 1897
7864 1898
7864 1899
7864 1900
7864 1901
7864 1902
7864 1903
7864 1904
7864 1905
7864 1906
7864 1907
7864 1908
7864 1909

0000'CD 05 3E
OF 0000'3A
C9 0871'D2
OF 0000'3A
3D D3 0A79'DA

787A 1910
787A 1911 01 3E
787C 1912 0000'32
787F 1913
787F 1914 00 00 21
7882 1915 0000'22
7885 1916
7885 1917 00'42'21
7888 1918 00'80'11
7888 1919 0000'CD
788E 1920
788E 1921 C5 46 00'00'21
77 18 3E

7896 1922
7896 1923 00'80'21
7899 1924 00'92'11
789C 1925 0000'CD
789F 1926
789F 1927 00'80'21
78A2 1928 00'00'11
78A5 1929 0000'CD
78A8 1930
78A8 1931 0000'CD
78A8 1932
78A8 1933 0000'CD
78AE 1934
78AE 1935 02F0'CD
78B1 1936
78B1 1937
78B1 1938 0000'3A
78B4 1939 OF
78B5 1940 089F'DA
78B8 1941
78B8 1942 B7
78B9 1943
78B9 1944 00'7F'21

B 4

Fiche 1 Frame B4

Sequence 40

ERROR2: WCS RAM DATA TEST FAILURE BELOW 16k.
ERROR3: WCS RAM DATA TEST FAILURE ABOVE 16k.

NOTE: ADDITIONAL DATA WILL GIVE WCS LOCATION BEING TESTED.

DEBUG: 1) CHECK LATCH ON BUS AD AND ENABLE SIGNAL 'WRT WWD REG H'
2) CHECK SIGNALS 'CS WE0 L','CS WE1 L' AND 'WCS WE2 L'
FROM THE REFRESH LOGIC
3) CHECK 'WRITE WCS B0','WRITE WCS B1','WRITE WCS B2'
FROM THE 8085 M1 LATCH
4) CHECK THE 'CS DOUT H' SIGNALS FEEDING THE CSR
5) CHECK 'PARAL LD CSR H' FOR PARALLEL LOAD OF CSR
6) CHECK THE RAM MEMORY FOR ERRORS

TEST5: HEAD <^X05>

MVI A,1 ;FIRST PART OF TEST 5.
STA CON_ERR_NUM

LXI H,0 ;POINT TO LOCATION ZERO IN WCS.
SHLD WCS_ADDRESS

LXI H,ONE_SHIFT ;MOVE A ONE THROUGH A ZERO FIELD
LXI D,TEMPB_DAT ; PATTERN TO TEMPB_DAT.
CALL MOVER_3

DO 24

LXI H,TEMPB_DAT ;MOVE THE PATTERN FROM TEMPB_DAT TO
LXI D,EXP_DAT ; EXP_DAT.
CALL MOVER_3

1\$: LXI H,TEMPB_DAT ;MOVE THE PATTERN INTO WCS_VALUE.
LXI D,WCS_VALUE
CALL MOVER_3

CALL WCS_WRITE ;WRITE TO WCS.
CALL WCS_READ ;READ FROM WCS.
CALL CHECK_5_6_ERROR ;COMPARE THE EXPECTED AND RECEIVED
;DATA AND PRINT ERROR IF ANY.

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 1\$;...JUMP TO BEGINNING OF ERROR LOOP.

ORA A ;CLEAR THE CARRY.

LXI H,TEMPB_SHIFT ;SHIFT THE TEMPB_DAT AREA TO MOVE THE

ZZ-ENKBA-2.0	7000	4
799A 2056	092F'C2	
799D 2057		
799D 2058	0000'CD	
79A0 2059	0000'3A	
79A3 2060	0F	
79A4 2061	00B7'DC	
79A7 2062		
79A7 2063	092F'C3	
79AA 2064		
79AA 2065		
79AA 2066	03 3E	
79AC 2067	0000'32	
79AF 2068		
79AF 2069	FF 3F 21	
79B2 2070	0000'22	
79B5 2071		
79B5 2072		
79B5 2073		
79B5 2074		
79B5 2075		
79B5 2076	3F 3E	
79B7 2077	0007'32	
79BA 2078		
79BA 2079	00'3B'21	
79BD 2080	00'80'11	
79C0 2081	0000'CD	
79C3 2082		
79C3 2083	C5 46 00'00'21 77 04 3E	
79CB 2084		
79CB 2085	00'80'21	
79CE 2086	00'92'11	
79D1 2087	0000'CD	
79D4 2088		
79D4 2089	00C7'CD	
79D7 2090		
79D7 2091	00'80'21	
79DA 2092	00'00'11	
79DD 2093	0000'CD	
79E0 2094		
79E0 2095	0000'CD	
79E3 2096		
79E3 2097	0000'CD	
79E6 2098		
79E6 2099	02F0'CD	
79E9 2100		
79E9 2101		
79E9 2102	0000'3A	
79EC 2103	0F	
79ED 2104	09D4'DA	
79F0 2105		
79F0 2106	B7	
79F1 2107		
79F1 2108	00'7F'21	
79F4 2109	0000'22	
79F7 2110	0000'CD	
79FA 2111		
79FA 2112	35 00'00'21 70 C1 09CB'C2	
7A03 2113		

E 4

```

Fiche 1 Frame E4 Sequence 43
JNZ      3$      ;IF NOT ZERO TEST NEXT WCS LOC.

CALL     GCVECT      ;SEE IF A CONTROL C HAS BEEN
LDA      CNTLC_TYPED ; TYPED.
RRC
CC       CON_SET_FOR_CO ;IF SO, GO TO MONITOR.

JMP      3$      ;TEST THE NEXT WCS LOCATION.

MVI      A,3      ;THIRD PART OF TEST 5.
STA      CON_ERR_NUM

LXI      H,<^XFF3F> ;LOAD ADDRESS OF FIRST 'ADDITIONAL'
SHLD     WCS_ADDRESS ; WCS LOCATION. WCS ABOVE 16K IS
                ; IMPLEMENTED USING 1K X 4 MEMORY
                ; CHIPS WHICH MUST BE TESTED USING
                ; A ONE THROUGH A ZERO STRING AND
                ; VISA VERSA.

MVI      A,<^X3F>   ;STOPE 64 IN LOOP_CNT.
STA      LOOP_CNT

LXI      H,ADD_WCS_DAT ;LOAD THREE BYTES OF ADDITIONAL WCS
LXI      D,TEMPB_DAT  ; DATA (ONE THROUGH ZERO FIELD FOUR
CALL     MOVER_3      ; BIT BLOCKS) INTO TEMPB_DAT.

DO       4

LXI      H,TEMPB_DAT ;MOVE THE DATA STRING INTO EXP_DAT.
LXI      D,EXP_DAT
CALL     MOVER_3

CALL     CHECK_RD      ;CALL BOOT CODE IF RD IS IN CONTROL

LXI      H,TEMPB_DAT ;MOVE THE DATA STRING INTO WCS_VALUE.
LXI      D,WCS_VALUE
CALL     MOVER_3

CALL     WCS_WRITE     ;WRITE TO WCS.

CALL     WCS_READ      ;READ FROM WCS.

CALL     CHECK_5_6_ERROR ;COMPARE THE EXPECTED AND RECEIVED
                ;DATA AND PRINT ERROR IF ANY.

LDA      ERROR_CON    ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC       8$          ;...JUMP TO BEGINNING OF ERROR LOOP.

ORA      A            ;CLEAR THE CARRY.

LXI      H,TEMPB_SHIFT ;SHIFT THE TEMPB DAT AREA TO SHIFT
SHLD     SHIFT_PNT    ; THE ONE THROUGH THE FIELD OF ZEROS.
CALL     SHIFTER

ENDDO

```

ZZ
.M
Sy

TE TE TE TE TE TE TE TE TE TE TE TE TE TT TT TT TU TU TU TU UP UP UP VE VN VS WC WC WC WC WC WC WC WC WC WC WR X-X-Y YB ZE ZF

```

ZZ-ENKBA-2.0 7000 4
7A03 2114 00'3E'21
7A06 2115 00'80'11
7A09 2116 0000'CD
7A0C 2117
7A0C 2118 C5 46 00'00'21
          77 04 3E
7A14 2119
7A14 2120 00'80'21
7A17 2121 00'92'11
7A1A 2122 0000'CD
7A1D 2123
7A1D 2124 00C7'CD
7A20 2125
7A20 2126 00'80'21
7A23 2127 00'00'11
7A26 2128 0000'CD
7A29 2129
7A29 2130 0000'CD
7A2C 2131
7A2C 2132 0000'CD
7A2F 2133
7A2F 2134 02F0'CD
7A32 2135
7A32 2136
7A32 2137 0000'3A
7A35 2138 0F
7A36 2139 0A1D'DA
7A39 2140
7A39 2141 37
7A3A 2142
7A3A 2143 00'7F'21
7A3D 2144 0000'22
7A40 2145 0000'CD
7A43 2146
7A43 2147 35 00'00'21
          70 C1 0A14'C2
7A4C 2148
7A4C 2149 00'00'21
          02 0E 00'0C'11
          0A5D'C2 0000'CD
7A5A 2150
7A5A 2151
7A5A 2152 0A77'C3
7A5C 2153
7A5D 2154
7A5D 2155
7A5D 2156 00'00'21
7A60 2157 0000'CD
7A63 2158
7A63 2159 00'07'21
7A66 2160 35
7A67 2161 09BA'C2
7A6A 2162
7A6A 2163 0000'CD
7A6D 2164 0000'3A
7A70 2165 0F
7A71 2166 00B7'DC
7A74 2167
7A74 2168
7A74 2169 09B5'C3

```

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

9\$:

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

F 4

Fiche 1 Frame F4

Sequence 44

```

LXI H,ADD WCS_DAT+3 ;LOAD THREE BYTES OF ADDITIONAL WCS
LXI D,TEMPB_DAT ; DATA (ZERO THROUGH ONE FIELD FOUR
CALL MOVER_3 ; BIT BLOCKS) INTO TEMPB_DAT.

```

DO 4

```

LXI H,TEMPB_DAT ;MOVE THE DATA STRING INTO EXP_DAT.
LXI D,EXP_DAT
CALL MOVER_3

```

CALL CHECK_RD ;CALL BOOT CODE IF RD IS IN CONTROL

```

LXI H,TEMPB_DAT ;MOVE THE DATA STRING INTO WCS_VALUE.
LXI D,WCS_VALUE
CALL MOVER_3

```

CALL WCS_WRITE ;WRITE TO WCS.

CALL WCS_READ ;READ FROM WCS.

CALL CHECK_5_6_ERROR ;COMPARE THE EXPECTED AND RECEIVED
;DATA AND PRINT ERROR IF ANY.

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...

RRC 9\$;...JUMP TO BEGINNING OF ERROR LOOP.

JC ;SET THE CARRY.

```

LXI H,TEMPB_SHIFT ;SHIFT THE TEMPB_DAT AREA TO SHIFT
SHLD SHIFT_PNT ; THE ZERO THROUGH A STRING OF ONES.
CALL SHIFTER

```

ENDDO

IFEQ WCS_ADDRESS,WCS_SIZE,2 ;HAS THE WCS POINTER EXCEEDED

; THE WCS SIZE.

JMP 10\$;TEST IS COMPLETED.

ENDIF

```

LXI H,WCS_ADDRESS ;INCREMENT THE WCS POINTER.
CALL INC_WORD

```

```

LXI H,LOOP_CNT ;POINT TO LOOP CNT.
DCR M ;DECREMENT LOOP CNT.
JNZ 7$ ;IF NOT ZERO TEST NEXT WCS LOC.

```

```

CALL GCVECT ;SEE IF A CONTROL C HAS BEEN
LDA CNTLC_TYPED ; TYPED.
RRC

```

CC CON_SET_FOR_CO ;IF SO, GO TO MONITOR.

JMP 6\$;TEST THE NEXT WCS ADDRESS AND

ZZ
.M
Ps

PS
--
.
CP

Ph
--
In
Co
Pa
Sy
Pa
Sy
Ps
Cr
As

Th
38
Th
47
15

Ma
--
SY

0

Th

/S

7A77 2170
7A77 2171
7A77 2172 3C D3

108:

TAIL

7A79 2173
7A79 2174
7A79 2175
7A79 2176
7A79 2177
7A79 2178
7A79 2179
7A79 2180
7A79 2181
7A79 2182
7A79 2183
7A79 2184
7A79 2185
7A79 2186
7A79 2187
7A79 2188
7A79 2189
7A79 2190
7A79 2191
7A79 2192
7A79 2193
7A79 2194
7A79 2195
7A79 2196
7A79 2197
7A79 2198
7A79 2199
7A79 2200
7A79 2201
7A79 2202
7A79 2203
7A79 2204
7A79 2205
7A79 2206
7A79 2207
7A79 2208
7A79 2209
7A79 2210
7A79 2211
7A79 2212
7A79 2213
7A79 2214
7A79 2215
7A79 2216
7A79 2217
7A79 2218
7A79 2219
7A79 2220
7A79 2221 0000'CD 06 3E
OF 0000'3A
C9 0A86'D2
OF 0000'3A
3D D3 0B83'DA

7A8F 2222
7A8F 2223 01 00 21
7A92 2224 0000'22
7A95 2225

TEST6

ABBREVIATED RAM MARCH PATTERN TEST

THE WCS RAM MARCH PATTERN TEST WRITES A BACKGROUND OF ZEROS AT EACH WCS LOCATION WHOSE ADDRESS HAS ONE BIT SET. THE TEST THEN GOES UP THROUGH MEMORY READING THE 0 BACKGROUND WRITING A 1 AND READING THE ONE BACK. THIS THEN LEAVES 1'S IN THE BACKGROUND AND THE TEST WILL START AT THE HIGH ADDRESS AND READ 1'S, WRITE 0'S, READ 0'S.

TEST STEPS:

- 1) SET ADDRESS IN UPC.
- 2) WRITE BACKGROUND OF ZEROS.
- 3) START AT LOCATION 1, READ AND COMPARE TO ZERO.
- 4) PRINT ERROR IF ANY.
- 5) WRITE 1, READ AND COMPARE TO 1.
- 6) PRINT ERROR IF ANY.
- 7) REPEAT 3-6 FOR ALL LOCATIONS (LOW TO HIGH).
- 8) STARTING AT TOP OF RAM, READ AND COMPARE TO ONE.
- 9) PRINT ERROR IF ANY.
- 10) WRITE 0 READ AND COMPARE TO 0.
- 11) PRINT ERROR IF ANY.
- 12) REPEAT 8-11 FOR ALL LOCATIONS (HIGH TO LOW).

ERROR1: WCS RAM MARCH TEST FAILURE ON FIRST PASS (UP).
ERRGR2: WCS RAM MARCH TEST FAILURE ON SECOND PASS (DOWN).

NOTE: ADDITIONAL DATA WILL GIVE THE LOCATION WHERE THE ERROR OCCURS.

DEBUG: 1) CHECK 'CS ADRS H','ROW ADRS STB L','COL ADRS STB L'
FOR THE ADDRESS BEING USED.
2) CHECK RAM MEMORY FOR ERRORS IN INTERNAL ADDRESS LOGIC.

TEST6:

HEAD <^X06>

LXI H,<^X0100> ;START AT WCS LOCATION 1.
SHLD WCS_ADDRESS

7A95	2226	00'41'21	7000	4
7A98	2227	00'00'11		
7A9B	2228	0000'CD		
7A9E	2229			
7A9E	2230	C5 46 00'00'21		
		77 0E 3E		
7AA6	2231			
7AA6	2232	0000'CD		
7AA9	2233			
7AA9	2234	B7		
7AAA	2235			
7AAA	2236	00'00'21		
7AAD	2237	0000'22		
7AB0	2238	0000'CD		
7AB3	2239			
7AB3	2240	00C7'CD		
7AB6	2241			
7AB6	2242	35 00'00'21		
		70 C1 0AA6'C2		
7ABF	2243			
7ABF	2244			
7ABF	2245	01 3E		
7AC1	2246	0000'32		
7AC4	2247			
7AC4	2248	01 00 21		
7AC7	2249	0000'22		
7ACA	2250			
7ACA	2251	C5 46 00'00'21		
		77 0E 3E		
7AD2	2252			
7AD2	2253	00'41'21		
7AD5	2254	00'92'11		
7AD8	2255	0000'CD		
7ADB	2256			
7ADB	2257	0000'CD		
7ADE	2258			
7ADE	2259	02F0'CD		
7AE1	2260			
7AE1	2261			
7AE1	2262	0000'3A		
7AE4	2263	0F		
7AE5	2264	0ADB'DA		
7AEB	2265			
7AEB	2266	00'45'21		
7AEB	2267	00'92'11		
7AEE	2268	0000'CD		
7AF1	2269			
7AF1	2270	00'45'21		
7AF4	2271	00'00'11		
7AF7	2272	0000'CD		
7AFA	2273			
7AFA	2274	0000'CD		
7AFD	2275			
7AFD	2276	0000'CD		
7B00	2277			
7B00	2278	02F0'CD		
7B03	2279			
7B03	2280			
7B03	2281	0000'3A		
7B06	2282	0F		

```

H 4
Fiche 1 Frame H4 Sequence 46
LXI H,ZERO_DAT ;LOAD ALL ZEROS INTO WCS_VALUE.
LXI D,WCS_VALUE
CALL MOVER_3

DO 14 ;WRITE TO 14 WCS LOCATIONS.

CALL WCS_WRITE ;WRITE A ZERO TO WCS.
ORA A ;CLEAR THE CARRY.
LXI H,WCS_SHIFT ;SHIFT THE ADDRESS FIELD.
SHLD SHIFT_PNT
CALL SHIFTER

CALL CHECK_RD ;CALL BOOT CODE IF UNDER RD CONTROL
ENDDO

MVI A,1 ;FIRST PART OF TEST 6.
STA CON_ERR_NUM

LXI H,<^X0100> ;START AT LOCATION 1.
SHLD WCS_ADDRESS

DO 14

LXI H,ZERO_DAT ;LOAD ALL ZEROS INTO EXP_DAT.
LXI D,EXP_DAT
CALL MOVER_3

CALL WCS_READ ;READ WCS.
CALL CHECK_5_6_ERROR ;COMPARE THE EXPECTED AND RECEIVED
;DATA AND PRINT ERROR IF ANY.
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 1$ ;...JUMP TO BEGINNING OF ERROR LOOP.

LXI H,ONE_DAT ;LOAD ALL ONES INTO EXP_DAT.
LXI D,EXP_DAT
CALL MOVER_3

LXI H,ONE_DAT ;LOAD ALL ONES INTO WCS_VALUE.
LXI D,WCS_VALUE
CALL MOVER_3

CALL WCS_WRITE ;WRITE TO WCS.
CALL WCS_READ ;READ FROM WCS.
CALL CHECK_5_6_ERROR ;COMPARE THE EXPECTED AND RECEIVED
;DATA AND PRINT ERROR IF ANY.
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC

```

ZZ-ENKBA-2.0 7000 4

```

7B07 2283 0AF1'DA *
7B0A 2284 *
7B0A 2285 B7 *
7B0B 2286 *
7B0B 2287 00'00'21 *
7B0E 2288 0000'22 *
7B11 2289 0000'CD *
7B14 2290 *
7B14 2291 00C7'CD *
7B17 2292 *
7B17 2293 35 00'00'21 *****
          70 C1 0AD2'C2

```

```

7B20 2294
7B20 2295
7B20 2296 02 3E
7B22 2297 0000'32
7B25 2298
7B25 2299 00 20 21
7B28 2300 0000'22
7B2B 2301
7B2B 2302 C5 46 00'00'21 *****
          77 0E 3E

```

```

7B33 2303
7B33 2304 00'45'21
7B36 2305 00'92'11
7B39 2306 0000'CD
7B3C 2307
7B3C 2308 0000'CD 3$:
7B3F 2309
7B3F 2310 02F0'CD
7B42 2311
7B42 2312
7B42 2313 0000'3A
7B45 2314 0F
7B46 2315 0B3C'DA
7B49 2316
7B49 2317
7B49 2318 00'41'21
7B4C 2319 00'92'11
7B4F 2320 0000'CD
7B52 2321
7B52 2322 00'41'21 4$:
7B55 2323 00'00'11
7B58 2324 0000'CD
7B5B 2325
7B5B 2326 0000'CD
7B5E 2327
7B5E 2328 0000'CD
7B61 2329
7B61 2330 02F0'CD
7B64 2331
7B64 2332
7B64 2333 0000'3A
7B67 2334 0F
7B68 2335 0B52'DA
7B6B 2336
7B6B 2337 B7
7B6C 2338
7B6C 2339 00'00'21
7B6F 2340 0000'22

```

I 4

Fiche 1 Frame 14 Sequence 47
;...JUMP TO BEGINNING OF ERROR LOOP.

```

JC 2$
ORA A ;CLEAR THE CARRY.
LXI H,WCS_SHIFT ;SHIFT THE ADDRESS FIELD.
SHLD SHIFT_PNT
CALL SHIFTER
CALL CHECK_RD ;CALL BOOT CODE IF UNDER RD CONTROL
ENDDO

```

```

MVI A,2 ;SECOND PART OF TEST 6.
STA CON_ERR_NUM

```

```

LXI H,<^X0020> ;START AT THE TOP OF WCS.
SHLD WCS_ADDRESS

```

```
DO 14
```

```

LXI H,ONE_DAT ;LOAD ALL ONES INTO EXP_DAT.
LXI D,EXP_DAT
CALL MOVER_3

```

```

CALL WCS_READ ;READ WCS.
CALL CHECK_5_6_ERROR ;COMPARE THE EXPECTED AND RECEIVED
;DATA AND PRINT ERROR IF ANY.
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 3$ ;...JUMP TO BEGINNING OF ERROR LOOP.

```

```

LXI H,ZERO_DAT ;LOAD ALL ZEROS INTO EXP_DAT.
LXI D,EXP_DAT
CALL MOVER_3

```

```

LXI H,ZERO_DAT ;LOAD ALL ZEROS INTO WCS VALUE.
LXI D,WCS_VALUE
CALL MOVER_3

```

```
CALL WCS_WRITE ;WRITE TO WCS.
```

```
CALL WCS_READ ;READ FROM WCS.
```

```
CALL CHECK_5_6_ERROR ;COMPARE THE EXPECTED AND RECEIVED
;DATA AND PRINT ERROR IF ANY.
```

```

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 4$ ;...JUMP TO BEGINNING OF ERROR LOOP.

```

```
ORA A ;CLEAR THE CARRY.
```

```

LXI H,WCS_SHIFT ;SHIFT THE ADDRESS FIELD.
SHLD SHIFT_PNT

```



```

ZZ-ENKBA-2.0 7000 4
7B72 2341 01E9'CD
7B75 2342
7B75 2343 00C7'CD
7B78 2344
7B78 2345 35 00'00'21
70 C1 0B33'C2
7B81 2346
7B81 2347 3C D3
7B83 2348
7B83 2349
7B83 2350
7B83 2351
7B83 2352
7B83 2353
7B83 2354
7B83 2355
7B83 2356
7B83 2357
7B83 2358
7B83 2359
7B83 2360
7B83 2361
7B83 2362
7B83 2363
7B83 2364
7B83 2365
7B83 2366
7B83 2367
7B83 2368
7B83 2369
7B83 2370
7B83 2371
7B83 2372
7B83 2373
7B83 2374
7B83 2375
7B83 2376
7B83 2377
7B83 2378
7B83 2379
7B83 2380
7B83 2381
7B83 2382
7B83 2383
7B83 2384
7B83 2385
7B83 2386
7B83 2387
7B83 2388
7B83 2389
7B83 2390
7B83 2391
7B83 2392
7B83 2393
7B83 2394
7B83 2395
7B83 2396
7B83 2397
7B83 2398
7B83 2399 07 3E

```

```

*
*
*
*
*****

```

J 4

Fiche 1 Frame J4

Sequence 48

CALL SHIFT_RT

CALL CHECK_RD ;CALL BOOT CODE IF UNDER RD CONTROL

ENDDO

TAIL

TEST7

CONTROL STORE PARITY TEST

THE CONTROL STORE PARITY TEST WILL TEST EACH CONTROL INPUT TO THE PARITY CHECKING CIRCUIT WITH A PATTERN OF ONE RIPPLED THROUGH A FIELD OF ZEROS AND A ZERO RIPPLED THROUGH A FIELD OF ONES. THIS WILL PROVE THAT EACH LINE IS CONNECTED. THEN THE TEST WILL LOAD PATTERNS INTO THE CSR WHICH WILL CAUSE BAD PARITY IN EACH OF THE PARITY CHECKERS.

TEST STEPS:

- 1) SHIFT PATTERN INTO CSR.
- 2) TURN ON PARITY.
- 3) PRINT ERROR IF ANY.
- 4) REPEAT 1 - 3 FOR ALL PATTERNS.

FOR THE TWO PARITY CHECKERS

- 5) EVEN AND EVEN.
- 6) PRINT ERROR IF ANY.
- 7) ODD AND ODD.
- 8) PRINT ERROR IF ANY.

ERROR1: PARITY CHECKING ERROR.
ERROR2: PARITY ERROR NOT FOUND.

NOTE: ADDITIONAL DATA WILL CONTAIN DATA SENT TO THE CSR.

- DEBUG:
- 1) CHECK THE LINES LEADING FROM THE CSR TO THE PARITY CHECKER 'CSR H'.
 - 2) CHECK THE OUTPUTS OF THE PARITY CHECKER (THE EVEN AND ODD OUTPUTS) FOR THE DATA GIVEN BY THE TEST AND THE GATES TO THE 'CS PARITY ERR H' SIGNAL.
 - 3) CHECK 'HALT ON PE L' FOR LOW VALUE ENABLING THE PARITY ERROR TO BE READ.

TEST7: MVI A,<^X07> ;TEST 7 INDICATOR.

ZZ-ENKBA-2.0 7000 4

```

7BE4 2457
7BE4 2458 22 D3
7BE6 2459
7BE6 2460 FB
7BE7 2461
7BE7 2462 C1
7BE8 2463
7BE8 2464 0341'CD
7BEB 2465
7BEB 2466 0C00'3A
7BEE 2467 0F
7BEF 2468 0BC1'DA
7BF2 2469
7BF2 2470 B7
7BF3 2471
7BF3 2472 00'7F'21
7BF6 2473 0000'22
7BF9 2474 0000'CD
7BFC 2475
7BFC 2476 00C7'CD
7BFF 2477
7BFF 2478 35 00'00'21
              70 C1 0BC1'C2

```

```

7C08 2479
7C08 2480
7C08 2481 00'46'21
7C08 2482 00'80'11
7C0E 2483 0000'CD
7C11 2484
7C11 2485 C5 46 00'00'21
              77 18 3E

```

```

7C19 2486
7C19 2487 00'80'21
7C1C 2488 00'00'11
7C1F 2489 0000'CD
7C22 2490
7C22 2491 0000'32 AF
7C26 2492 0000'CD
7C29 2493
7C29 2494 00'73'21
7C2C 2495 4C 22 11
7C2F 2496 0000'CD
7C32 2497
7C32 2498 A0 D3
7C34 2499
7C34 2500 00
7C35 2501
7C35 2502
7C35 2503 A1 D3
7C37 2504
7C37 2505 0C43'C3
7C3A 2506
7C3A 2507 A1 D3
7C3C 2508
7C3C 2509 22 D3
7C3E 2510
7C3E 2511 FB
7C3F 2512
7C3F 2513 C1
7C40 2514

```

```

7C40 2514

```

L 4

Fiche 1 Frame L4

Sequence 50

```

OUT    CLRSS           ;CLEAR SINGLE STEP.
EI      ;ENABLE FURTHER INTERRUPTS.
POP     B              ;CLEAN UP STACK.
CALL    TEST7_ERROR    ;PRINT AN ERROR MESSAGE.
LDA     ERROR_CON      ;IF THE ERROR LOOPING FLAG IS SET...
RRC     1$             ;...JUMP TO BEGINNING OF ERROR LOOP.
JC      1$
ORA     A              ;CLEAR THE CARRY.
LXI     H,TEMPB_SHIFT  ;SHIFT THE ONE THROUGH THE ZERO FIELD.
SHLD    SHIFT_PNT
CALL    SHIFTER
CALL    CHECK_RD       ;CALL BOOT CODE IF UNDER RD CONTROL
ENDDO

```

```

LXI     H,ZERO_SHIFT   ;LOAD THE SECOND TEST PATTERN (ZERO
LXI     D,TEMPB_DAT    ; THROUGH A STRING OF ONES) INTO
CALL    MOVER_3         ; TEMPB_DAT.

```

DO 24

```

LXI     H,TEMPB_DAT    ;LOAD THE TEST PATTERN INTO CSR_VALUE.
LXI     D,CSR_VALUE
CALL    MOVER_3

```

```

CLEAR   PARITY_ON      ;DO NOT CALCULATE PARITY WHEN LOADING
CALL    LOAD_CSR       ; THE TEST PATTERN INTO THE CSR.

```

```

LXI     H,TEST7_JUMP+3 ;PREPARE TO RETURN FROM A PARITY ERROR.
LXI     D,PARITY_VECTOR
CALL    MOVER_3

```

```

OUT     ENBPARG        ;SET ENABLE PARITY ERROR.

```

```

NOP      ;ALLOW SUFFICIENT TIME FOR A PARITY
          ; ERROR TO BE DETECTED.

```

```

OUT     DISPAR         ;DISABLE PARITY ERROR.

```

```

JMP     6$             ;SKIP ERROR PROCEDURE.

```

```

OUT     DISPAR         ;DISABLE PARITY ERROR.

```

```

OUT     CLRSS          ;CLEAR SINGLE STEP.

```

```

EI      ;ENABLE FURTHER INTERRUPTS.

```

```

POP     B              ;CLEAN UP STACK.

```

```

ZZ-ENKBA-2.0 7000 4
7C40 2515 0341'CD *
7C43 2516 *
7C43 2517 0000'3A * 6$:
7C46 2518 0F *
7C47 2519 0C19'DA *
7C4A 2520 *
7C4A 2521 37 *
7C4B 2522 *
7C4B 2523 00'7F'21 *
7C4E 2524 0C00'22 *
7C51 2525 0000'CD *
7C54 2526 *
7C54 2527 00C7'CD *
7C57 2528 *
7C57 2529 35 00'00'21 *****
          70 C1 0C19'C2

7C60 2530
7C60 2531
7C60 2532 02 3E
7C62 2533 0000'32
7C65 2534
7C65 2535 00'41'21 7$:
7C68 2536 00'80'11
7C6B 2537 0000'CD
7C6E 2538
7C6E 2539 00'41'21
7C71 2540 00'00'11
7C74 2541 0000'CD
7C77 2542
7C77 2543 0000'32 AF
7C7B 2544
7C7B 2545 0000'CD
7C7E 2546
7C7E 2547 00'76'21
7C81 2548 4C 22 11
7C84 2549 0000'CD
7C87 2550
7C87 2551 A0 D3
7C89 2552
7C89 2553 00
7C8A 2554
7C8A 2555
7C8A 2556 A1 D3
7C8C 2557
7C8C 2558 0341'CD
7C8F 2559
7C8F 2560 0000'3A
7C92 2561 0F
7C93 2562 0C65'DA
7C96 2563
7C96 2564 0CA6'C3
7C99 2565
7C99 2566
7C99 2567
7C99 2568 A1 D3 8$:
7C9B 2569
7C9B 2570 22 D3
7C9D 2571
7C9D 2572 FB
7C9E 2573

```

```

M 4
Fiche 1 Frame M4 Sequence 51
CALL TEST7_ERROR ;PRINT AN ERROR.
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 4$ ;...JUMP TO BEGINNING OF ERROR LOOP.
STC ;SET THE CARRY.
LXI H,TEMPB_SHIFT ;SHIFT THE ZERO THROUGH THE STRING
SHLD SHIFT_PNT ; OF ONES.
CALL SHIFTER
CALL CHECK_RD ;CALL BOOT CODE IF UNDER RD CONTROL
ENDDO

MVI A,2 ;SECOND PART OF TEST 7.
STA CON_ERR_NUM
LXI H,ZERO_DAT ;LOAD ALL ZEROS INTO TEMPB_DAT.
LXI D,TEMPB_DAT
CALL MOVER_3
LXI H,ZERO_DAT ;LOAD ALL ZEROS INTO CSR_VALUE.
LXI D,CSR_VALUE
CALL MOVER_3
CLEAR PARITY_ON ;DO NOT CALCULATE PARITY WHEN LOADING
CALL LOAD_CSR ; THE TEST PATTERN INTO CSR.
LXI H,TEST7_JUMP+6 ;PREPARE TO RETURN FROM A PARITY ERROR.
LXI D,PARITY_VECTOR
CALL MOVER_3
OUT ENBPOR ;ENABLE PARITY ERROR.
NOP ;ALLOW SUFFICIENT TIME FOR A PARITY
; ERROR TO BE DETECTED.
OUT DISPAR ;DISABLE PARITY ERROR.
CALL TEST7_ERROR ;PRINT AN ERROR.
LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC 7$ ;...JUMP TO BEGINNING OF ERROR LOOP.
JMP 9$ ;SKIP THE NEXT FEW LINES OF CODE
; WHICH HANDLE A RETURN FROM PARITY
; ERROR.
OUT DISPAR ;DISABLE PARITY ERROR.
OUT CLRSS ;CLEAR SINGLE STEP.
EI ;ENABLE FURTHER INTERRUPTS.

```

22-ENKBA-2.0	7000	4
7C9E 2574	C1	
7C9F 2575		
7C9F 2576	0000'3A	
7CA2 2577	OF	
7CA3 2578	0C65'DA	
7CA6 2579		
7CA6 2580		
7CA6 2581		
7CA6 2582	00'6D'21	
7CA9 2583	0C'80'11	
7CAC 2584	0000'CD	
7CAF 2585		
7CAF 2586	00'6D'21	
7CB2 2587	00'00'11	
7CB5 2588	0000'CD	
7CB8 2589		
7CB8 2590	0000'32	AF
7CBC 2591		
7CBC 2592	0000'CD	
7CBF 2593		
7CBF 2594	00'79'21	
7CC2 2595	4C 22 11	
7CC5 2596	0000'CD	
7CC8 2597		
7CC8 2598	A0 D3	
7CCA 2599		
7CCA 2600	00	
7CCB 2601		
7CCB 2602		
7CCB 2603	A1 D3	
7CCD 2604		
7CCD 2605	0341'CD	
7CD0 2606		
7CD0 2607	0000'3A	
7CD3 2608	OF	
7CD4 2609	0CA6'DA	
7CD7 2610		
7CD7 2611	0CE7'C3	
7CDA 2612		
7CDA 2613		
7CDA 2614	A1 D3	
7CDC 2615		
7CDC 2616	22 D3	
7CDE 2617		
7CDE 2618	FB	
7CDF 2619		
7CDF 2620	C1	
7CE0 2621		
7CE0 2622	0000'3A	
7CE3 2623	OF	
7CE4 2624	0CA6'DA	
7CE7 2625		
7CE7 2626	00'00'21	
7CEA 2627	4C 22 11	
7CED 2628	0000'CD	
7CF0 2629		
7CF0 2630	3C D3	
7CF2 2631		
7CF2 2632	00	
7CF3 2633		

```

N 4
Fiche 1 Frame N4
Sequence 52

POP B ;CLEAN UP STACK.

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC ;...JUMP TO BEGINNING OF ERROR LOOP.
JC 7$

LXI H,TEST7_DAT ;MOVE THE FOURTH TEST PATTERN (BOTH
LXI D,TEMPB_DAT ; PARITY CHECKERS ODD) INTO
CALL MOVER_3 ; TEMPB_DAT.

LXI H,TEST7_DAT ;MOVE THE FOURTH TEST PATTERN INTO
LXI D,CSR_VALUE ; CSR_VALUE.
CALL MOVER_3

CLEAR PARITY_ON ;DO NOT CALCULATE PARITY WHEN LOADING
CALL LOAD_CSR ; THE TEST PATTERN INTO CSR.

LXI H,TEST7_JUMP+9 ;PREPARE TO RETURN FROM A PARITY ERROR.
LXI D,PARITY_VECTOR
CALL MOVER_3

OUT ENBPARG ;ENABLE PARITY ERROR.

NOP ;ALLOW SUFFICIENT TIME FOR A PARITY
; ERROR TO BE DETECTED.

OUT DISPAR ;DISABLE PARITY ERROR.

CALL TEST7_ERROR

LDA ERROR_CON ;IF THE ERROR LOOPING FLAG IS SET...
RRC ;...JUMP TO BEGINNING OF ERROR LOOP.
JC 9$

JMP TEST7_CLEANUP

OUT DISPAR ;DISABLE PARITY ERROR.

OUT CLRSS ;CLEAR SINGLE STEP.

EI ;ENABLE FURTHER INTERRUPTS.

POP B ;CLEAN UP THE STACK.

LDA ERROR_CON
RRC
JC 9$

LXI H,PARITY_JUMP ;RETURN ORIGINAL VALUE TO PARITY VECTOR
LXI D,PARITY_VECTOR
CALL MOVER_3

OUT CLRLIT

NOP ;END OF TEST. LOCATION FOR JUMPING TO
; ON SKIP TEST.

```

```

7CF3 2634
7CF3 2635
7CF3 2636
7CF3 2637
7CF3 2638
7CF3 2639
7CF3 2640
7CF3 2641
7CF3 2642
7CF3 2643
7CF3 2644
7CF3 2645
7CF3 2646
7CF3 2647
7CF3 2648
7CF3 2649
7CF3 2650
7CF3 2651
7CF3 2652
7CF3 2653
7CF3 2654
7CF3 2655
7CF3 2656
7CF3 2657
7CF3 2658
7CF3 2659
7CF3 2660
7CF3 2661
7CF3 2662
7CF3 2663
7CF3 2664
7CF3 2665
7CF3 2666
7CF3 2667
7CF3 2668
7CF3 2669
7CF3 2670
7CF3 2671
7CF3 2672
7CF3 2673
7CF3 2674
7CF3 2675
7CF3 2676
7CF3 2677
7CF3 2678 0000'CD 08 3E
              0F 0000'3A
              C9 0D00'D2
              0F 0000'3A
              3D D3 0D67'DA
7D09 2679
7D09 2680 00'4C'21
7D0C 2681 0112'CD
7D0F 2682
7D0F 2683 0041'3A
7D12 2684
7D12 2685 038E'CD
7D15 2686
7D15 2687 0D0F'DA
7D18 2688
7D18 2689

```

TEST8

2901 DATA PATH TEST (D TO Y)

DATA WILL BE SENT TO THE 2901 OVER THE 8 BIT DATA PATH TO THE D-BUS LATCH. A JUMP INSTRUCTION WILL BE PUT INTO THE CSR TO CAUSE THE DATA TO BE PASSED DIRECTLY THROUGH THE 2901. THE Y-BUS LATCH WILL BE CLOCKED AND THEN READ BY THE CONSOLE. THE DATA USED WILL BE ALL 1'S, ALL 0'S, ZERO RIPPLED THROUGH ONES AND ONE RIPPLED THROUGH ZEROS.

TEST STEPS:

- 1) WRITE DATA INTO D-BUS LATCH.
- 2) PUT JUMP INTO CSR.
- 3) CLOCK Y-BUS LATCH.
- 4) READ Y-BUS LATCH TO CONSOLE AND COMPARE DATA.
- 5) PRINT ERROR IF ANY.
- 6) REPEAT 1 - 5 FOR EACH DATA PATTERN.

ERROR1: 2901 DATA PATH TEST ERROR (D TO Y).

- DEBUG:
- 1) CHECK 'SRC CTL H' FOR 'D.O' = 7.
 - 2) CHECK 'ALU CTL H' CODE FOR 'R.OR.S' = 3.
 - 3) CHECK 'DEST CTL H' FOR 'NOP' = 1.
 - 4) CHECK 'CONS BUS H' BIT 0 - 7.
 - 5) CHECK THE D-BUS LATCH 'READ CONSOLE L' SIGNAL.
 - 6) CHECK THAT THE LOWER 8 BITS OF THE D-BUS INFORMATION REACHES THE 2901.
 - 7) CHECK THE LOWER 8 BITS OF THE Y-BUS AND THE Y-BUS LATCH 'WRITE CRR L' SIGNAL.
 - 8) CHECK 'SRC CTL H,DEST CTL H,ALU CTL H' FOR CSR 22 - 14.

TEST8:

HEAD <^X08>

1\$:

```

LXI    H,MOV_CCR_WRO ;LOAD THE CSR WITH A MOVE INSTRUCTION
CALL   LD_CSR_INST   ; USED TO PASS DATA THROUGH THE ALU.

LDA     ZERO_DAT      ;LOAD ALL ZEROS INTO A AND INTO

CALL    TEST8_1_0      ;

JC      1$             ;....JUMP TO BEGINNING OF ERROR LOOP.

```

22-ENKBA-2.0 7000 4

7D18 2690
7D18 2691 0045'3A
7D1B 2692
7D1B 2693 038E'CD
7D1E 2694
7D1E 2695 0D18'DA
7D21 2696
7D21 2697
7D21 2698
7D21 2699 0G44'3A
7D24 2700 007D'32
7D27 2701
7D27 2702 C5 46 00'00'21
77 08 3E
7D2F 2703
7D2F 2704 03B0'CD
7D32 2705
7D32 2706 B7
7D33 2707 007D'3A
7D36 2708 07
7D37 2709 007D'32
7D3A 2710
7D3A 2711 35 00'00'21
70 C1 0D2F'C2
7D43 2712
7D43 2713
7D43 2714
7D43 2715 0048'3A
7D46 2716 007D'32
7D49 2717
7D49 2718 C5 46 00'00'21
77 08 3E
7D51 2719
7D51 2720 03B0'CD
7D54 2721
7D54 2722 37
7D55 2723 007D'3A
7D58 2724 07
7D59 2725 007D'32
7D5C 2726
7D5C 2727 35 00'00'21
70 C1 0D51'C2
7D65 2728
7D65 2729 3C D3
7D67 2730
7D67 2731
7D67 2732
7D67 2733
7D67 2734
7D67 2735
7D67 2736
7D67 2737
7D67 2738 0000'CD 51 3E
0F 0000'3A
C9 0D74'D2
0F 0000'3A
3D D3 0D7F'DA
7D7D 2739
7D7D 2740 3C D3
7D7F 2741

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

2\$:

C 5

Fiche 1 Frame C5

Sequence 54

LDA ONE_DAT ;LOAD ALL ONES INTO A AND INTO
CALL TEST8_1_0 ;
JC 2\$;...JUMP TO BEGINNING OF ERROR LOOP.

LDA ONE_SHIFT+2 ;MOVE THE THIRD TEST PATTERN (ONE
STA TEMP_A_DAT ; THROUGH ZEROS) TO TEMP_A_DAT.
DO 8

CALL TEST8_SHIFT ;
ORA A ;CLEAR THE CARRY.
LDA TEMP_A_DAT ;ROTATE THE TEST PATTERN TO THE LEFT
RLC ; TO SHIFT THE ONE THROUGH THE ZERO
STA TEMP_A_DAT ; FIELD.

ENDDO

LDA ZERO_SHIFT+2 ;MOVE THE FOURTH TEST PATTERN (ZERO
STA TEMP_A_DAT ; THROUGH ONES) TO TEMP_A_DAT.

DO 8

CALL TEST8_SHIFT ;
STC ;SET THE CARRY.
LDA TEMP_A_DAT ;ROTATE THE TEST PATTERN TO THE LEFT
RLC ; TO SHIFT THE ZERO THROUGH THE ONE
STA TEMP_A_DAT ; FIELD.

ENDDO

TAIL

: LAST TEST FOR EVERY CONSOL_SECTION....FORCES END OF SECTION
:*****

END_TEST: HEAD <^X51>

TAIL

.MAIN.
Symbol table

```

SPSW      = 00000006
SSP       = 00000006
A         = 00000007
ADDR_16K  = 0000000A R    02
ADD_WCS_DAT = 0000003B R    02
ALLOWP    = 00000003
APTFLG    = 00000004
AUTO      = 00000005
B         = 00000000
BOOTEN    = 00000007
BOOTF     = 00000001
C         = 00000001
CHECK_5_6_ERROR = 000002F0 P    02
CHECK_RD  = 000000C7 R    02
CLRACK    = 000000A9
CLRACL    = 00000033
CLRATN    = 000000AB
CLRBSY    = 0000002E
CLRCLK    = 00000020
CLRCSK    = 00000024
CLRCSR    = 00000038
CLRDCL    = 00000031
CLRHLT    = 000000AF
CLRLIT    = 0000003C
CLRMCI    = 00000029
CLRMCK    = 0000002B
CLRPFI    = 000000AD
CLRSS     = 00000022
CLRTIM    = 000000A5
CLRTMR    = 0000002C
CLRUPC    = 000000A8
CLRUPK    = 00000026
CLRWCK    = 00000036
CNT       = 0000081F
CNTA      = 00000FA0
CNTLC_TYPED = ***** X    02
COMPARE   = ***** X    02
CON_ADD_DAT = ***** X    02
CON_BEGIN_TEST = ***** X    02
CON_ERROR = ***** X    02
CON_ERR_NUM = ***** X    02
CON_EXP_DAT = ***** X    02
CON_MOD_DAT = ***** X    02
CON_MSK_DAT = ***** X    02
CON_REC_DAT = ***** X    02
CON_SET_FOR_CO = 000000B7 R    02
CPATTN    = 00000083
CPUACK    = 00000082
CR        = ***** X    02
CSR07     = 00000085
CSR15     = 00000086
CSR23     = 00000087
CSR_SHIFT = ***** X    02
CSR_VALUE = ***** X    02
D         = 00000002
DCLO      = 00000002
DISMEM    = 000000A7

```

```

DISPAR    = 000000A1
DOLOOP    = ***** X    02
E         = 00000003
ENBMEM    = 000000A6
ENBPAT    = 000000A0
END_TEST  = 000000D7 R    02
END_TEST7 = 000000CF R    02
ENTRY     = 00000000 R    02
EOS_FLG   = ***** X    02
ERROR_4_1ST = 00000202 R    02
ERROR_4_2ND = 00000235 R    02
ERROR_CON = ***** X    02
EXEC_INST = 0000010A R    02
EXP2_DAT  = 000000AE R    02
EXP2_SHIFT = 000000AD R    02
EXP_DAT   = 00000092 R    02
EXP_SHIFT = 00000091 R    02
FILE_NAME = 00000005 R    02
GCVECT    = ***** X    02
H         = 00000004
HEAD      = 00000000
HEADER_B  = 00000010 R    02
HEX_3     = ***** X    02
HEX_BUF   = ***** X    02
HLTFLG    = 00000002
INC_WORD  = ***** X    02
INITH     = 00000035
INITL     = 00000034
ITCSR     = 0000001D
ITDB      = 0000001C
JMP_CONTROL = 00000004
JMP_INST  = 00000049 R    02
JMP_OP_CODE = 00000008
L         = 00000005
LD_CSR_INST = 00000112 R    02
LITERAL   = 00000001
LOAD_CSR  = ***** X    02
LOAD_MOD_DAT = 000001F9 R    02
LOAD_UPC  = ***** X    02
LOOP_CNT  = 00000007 R    02
LVL       = 00000000
M         = 00000006
MACSTK1   = 0000081F
MACSTK2   = 0000081D
MACSTK3   = 00000804
MAKE_NEXT_ADD = 000000D7 R    02
MCPU_A    = ***** X    02
MIPFLG    = 000040D9
MISC2     = 00000001
MOVER     = ***** X    02
MOVER_3   = ***** X    02
MOVER_4   = ***** X    02
MOV_CTR_WRO = 0000004C R    02
MWCS_A    = ***** X    02
NA_EXP_REC = ***** X    02
NA_OTHER  = ***** X    02
NOP_CSR   = ***** X    02

```

```

OKV15     = 00000007
ONE_DAT   = 00000045 R    02
ONE_SHIFT = 00000042 R    02
PARITY_JUMP = ***** X    02
PARITY_ON = ***** X    02
PARITY_VECTOR = 00004C22
POWER_VECTOR = 00004C25
PRINT_HEX = ***** X    02
PRINT_STRING = ***** X    02
READ      = 00000080
READJ1    = 00000003
READJ2    = 00000004
REC2_DAT  = 000000B3 R    02
REC2_SHIFT = 000000B2 R    02
REC_DAT   = 000000A0 R    02
REC_SHIFT = 0000009F R    02
REMDIS    = 00000006
SAVE_WCS_ADD = 00000008 R    02
SETACK    = 000000A8
SETACL    = 00000032
SETATN    = 000000AA
SETBSY    = 0000002F
SETCLK    = 00000021
SETCSK    = 00000025
SETCSR    = 00000039
SETDCL    = 00000030
SETHLT    = 000000AE
SETLIT    = 0000003D
SETMCI    = 00000028
SETMCK    = 0000002A
SETPFI    = 000000AC
SETSS     = 00000023
SETTIM    = 000000A4
SETTMR    = 0000002D
SETUPC    = 000000A9
SETUPK    = 00000027
SETWCK    = 00000037
SET_FOR_CONT = ***** X    02
SHIFTER   = ***** X    02
SHIFT_PNT = ***** X    02
SHIFT_RT  = 000001E9 R    02
SKIP_TEST = ***** X    02
SYM       = 0000081F
TEMPA_DAT = 0000007D R    02
TEMPA_SHIFT = 0000007C R    02
TEMPB_DAT = 00000080 R    02
TEMPB_SHIFT = 0000007F R    02
TEMPC_DAT = 00000084 R    02
TEMPC_SHIFT = 00000083 R    02
TEST1     = 000003D8 R    02
TEST1_DAT = 0000004F R    02
TEST1_ERROR = 00000250 R    02
TEST2     = 000004E9 R    02
TEST2_DAT = 00000057 R    02
TEST2_DAT2 = 00000065 R    02
TEST2_DAT3 = 00000069 R    02
TEST2_ERR4 = 000002A3 R    02

```


ZZ-ENKBA-2.0 Symbol table
MAIN.
Symbol table

TEST2_ERROR	0000025C	R	02
TEST3	000006FB	R	02
TEST3_4_ERROR	000002C9	R	02
TEST4	00000779	R	02
TEST5	00000864	R	02
TEST5_6_ERROR	00000302	R	02
TEST6	00000A79	R	02
TEST7	00000B83	R	02
TEST7_CLEANUP	00000CE7	R	02
TEST7_DAT	0000006D	R	02
TEST7_ERROR	00000341	R	02
TEST7_JUMP	00000070	R	02
TEST7_PAR_ERR	00000BA4	R	02
TEST8	00000CF3	R	02
TEST8_1_0	0000038E	R	02
TEST8_ERROR	00000362	R	02
TEST8_SHIFT	00000380	R	02
TTCR	= 0000004B		
TTDB	= 00000048		
TTMODE	= 0000004A		
TTSR	= 00000049		
TUCR	= 00000047		
TUDB	= 00000044		
TUMODE	= 00000046		
TUSR	= 00000045		
UPC14	= 00000084		
UPC14A	= 00000000		
UPC_VALUE	*****	X	02
VERSION	00000003	R	02
VNORML	= 000000A3		
VSHIFT	= 000000A2		
WCSB0	= 00000008		
WCSB1	= 00000009		
WCSB2	= 0000000A		
WCS_ADDRESS	*****	X	02
WCS_ADDR_PNT	0000000E	R	02
WCS_READ	*****	X	02
WCS_SHIFT	*****	X	02
WCS_SIZE	0000000C	R	02
WCS_SIZER	00000145	R	02
WCS_SIZER_ADDR	00000031	R	02
WCS_SIZE_P	000001D2	R	02
WCS_VALUE	*****	X	02
WCS_WRITE	*****	X	02
WRITE	= 000000CC		
X_CLR_PAGE	*****	X	02
X_SET_PAGE	*****	X	02
Y	= 00000044		
YBUSRD	= 000000EC		
ZERO_DAT	00000041	R	02
ZERO_SHIFT	00000046	R	02

E 5
7-OCT-1982
Fiche 1 Frame E5
Sequence 56
7-OCT-1982 12:37:05 VAX-11 Macro V03-00
7-OCT-1982 12:34:36 DRB1:[ERICCHETTI.WCS]ENKBA.MAC;201 (1)
Page 92

ZZ-ENKBA-2.0 Psect synopsis
.MAIN.
Psect synopsis

F 5
7-OCT-1982

Fiche 1 Frame F5

Sequence 57

7-OCT-1982 12:37:05 VAX-11 Macro V03-00 Page 93
7-OCT-1982 12:34:36 DRB1:[RICCHETTI.WCS]ENKBA.MAC;201 (1)

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS :	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK :	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
CPU1	00000D7F (3455.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	23	00:00:00.05	00:00:00.24
Command processing	34	00:00:00.24	00:00:01.02
Pass 1	1429	00:00:35.16	00:00:51.78
Symbol table sort	4	00:00:00.27	00:00:00.53
Pass 2	1412	00:00:13.17	00:00:33.31
Symbol table output	30	00:00:00.15	00:00:00.29
Psect synopsis output	5	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	2941	00:00:49.08	00:01:27.28

The working set limit was 1000 pages.
384428 bytes (751 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 222 non-local and 118 local symbols.
4765 source lines were read in Pass 1, producing 51 object records in Pass 2.
151 pages of virtual memory were used to define 150 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
SYS\$SY\$ROOT:[SYSLIB]STARLET.MLB;1	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

/SHOW=(BIN)/LIST=ENKBA.LIS/OBJECT=ENKBA.OBJ 8085MAC.MAR+MACDEF.MAC+[JENKBA.MAC

ZZ-ENKBA-2.0 Directory

G 5
7-OCT-1982

Fiche 1 Frame G5

Sequence 58

Fiche	Frame	Sequence	
1	B1	1	Documentation
1	H1	7	7000 4
1	D5	55	Symbol table
1	F5	57	Psect synopsis
1	G5	58	Directory

7-OCT-1982
7-OCT-1982
7-OCT-1982